International Journal on Advanced Science Engineering Information Technology

Vol.6 (2016) No. 5
ISSN: 2088-5334

A Survey on Adaptation Strategies for Mutation and Crossover Rates of Differential Evolution Algorithm

Dhanya M Dhanalakshmy[#1], Pranav P[#2], Jeyakumar G[#3]

[#] *Department of Computer Science and Engineering, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, Amrita University, India*
E-mail: [1]md_dhanya@cb.amrita.edu; [2]pranavprakash20@gmail.com; [3]g_jeyakumar@cb.amrita.edu

***Abstract*— Differential Evolution (*DE*), the well-known optimization algorithm, is a tool under the roof of Evolutionary Algorithms (EAs) for solving non-linear and non-differential optimization problems. *DE* has many qualities in its hand, which are attributing to its popularity. *DE* also known for its simplicity in solving the given problem with few control parameters: the population size (NP), the mutation rate (F) and the crossover rate (C_r). To avoid the difficulty involved in setting of suitable values for *NP*, *F* and C_r many parameter adaptation strategies are proposed in the literature. This paper is to present the working principle of the parameter adaptation strategies of *F* and C_r. The adaptation strategies are categorized based on the logic used by the authors, and clear insights about all the categories are presented.**

***Keywords*— Differential Evolution; Parameter Adaptation; Mutation Rate; Crossover Rate.**

I. INTRODUCTION

Differential Evolution (*DE*) (proposed by Storn and Price [1],[2], a population based stochastic search method, is a very powerful algorithm in the repository of Evolutionary Algorithms (*EAs*). The performance efficacy of *DE*, comparing with other *EAs*, for solving real time and benchmarking problems which are non-linear, complex and high dimensional over continuous domain has been well proved in its literature [3]. The algorithmic structure of *DE* is similar to other *EAs*. However, unlike other *EAs*, *DE* uses very few control parameters: the population size (*NP*), the mutation rate (*F*) and the crossover rate (C_r). The efficiency and accuracy of *DE* algorithm is more sensitive to the values chosen for these few parameters.

The successful convergence of *DE* to the global optimum solution, in its evolutionary search for solving the given problem, is largely depend on suitable selection of values for these control parameters. Finding the suitable values for these control parameters, before starting the search, is a difficult task as it will differ from problem to problem. A poor choice of these values will result in the poor accuracy of the algorithm which is not acceptable. There is no single perfect method or standard available for selecting values for these control parameters. Hence, the process of tuning these control parameters along with the search became an attractive area of research for the researchers' community working in *DE*. This results numerous adaptation strategies proposed for *NP*, *F* and C_r in *DE* literature. Subsequently, this has become a challenge to the practitioners, researchers and users of *DE* to choose right adaptation strategies for each of the control parameter to solve the problem at their hand. Resolving this challenge is taken as the aim of this paper.

The objective of this paper is to provide the readers with brief insight about various adaptation strategies proposed by researchers for adapting *F* and C_r. It is obvious that the number of researchers working in *DE*, particularly in the parameter adaptation of *DE*, is increasing day after the other. This paper is intended to provide them with summary of various adaptation strategies exist in *DE* literature for tuning *F* and C_r.

II. DIFFERENTIAL EVOLUTION ALGORITHM

For a search method to be efficient and reliable, it has to cover the entire search space. Differential Evolution starts its search of global solution for the given optimization problem, with randomly selected *NPD*-dimensional population vectors (individuals/candidates). The initial population is chosen in such a way that the individuals are initialized randomly in order to cover the entire search space. The population vector is represented as $X_{i,G} = \{x^1_{i,G}, x^2_{i,G} \ldots x^D_{i,G}\}$, *i* ranges from *1* to *NP*, *G* represents the generation and *D* represents the number of parameters for each individual (ie, dimension of the problem).

The three evolutionary processes involved in *DE* are mutation, crossover and selection. Among these the mutation and crossover are called variation operators, which brings changes in the population by altering the values of the components of the individuals in the population. The changes made by these operators create new candidates in the population, thus increasing the diversity of the population. Hence they attributed to exploration phase of the search. On the other hand, the selection process selects the best candidate from a set of candidates. Thus it is for the exploitation phase of the search.

At first the mutation process takes place. From the initial population the mutation process generates a mutated population. This process is termed as *Differential Mutation* in *DE*. The mutation process chooses three random candidates (say C_1, C_2 and C_3) from the population, and generates a mutant vector (*MV*) as

$$MV = C_1 + F * (C_2 - C_3) \quad (1)$$

where *F*- mutation rate or scaling factor. In Equ (1), the scaled difference of C_2 and C_3 is added to C_1 (also known as base vector). There exist many ways to choose the base vector and the other pair of vectors for mutation. Based on that, there are many mutation strategies available for *DE*. The critical parameter in the mutation process is the scaling factor *F*. One mutant vector is generated for each vector population (also known as target vector (*TaV*)) in the current, which results mutated population with *NP* mutant vectors.

Secondly, the *crossover* process generates the trial vector (*TrV*) population. This process recombines each of the *TaV* in the current generation with its corresponding *MV* to produce the *TrV*. The values from the parameters of *TaV* and *MV* are used to generate a *TrV*. The crossover process results one *TrV* for each *TaV*. The *crossover* process determines how much information the trial vector (child) inherits from its parents (target and mutant vectors).This is determined by the control parameter called the *crossover rate* (C_r). The most common two crossover strategies of *DE* are *binomial crossover* and *exponential crossover*. The equation for binomial crossover is given in equation (2). The crossover process also repeated for all the pair of target and the corresponding mutant vector, which results a population of *NP* trial vectors.

$$TrV_i = \begin{cases} MV_i & if\, C_r \geq Rand() \\ TaV_i & Otherwise \end{cases} \quad (2)$$

Next, a *selection* process is carried out between each target vector in the current population and their corresponding trial vectors. *DE* uses one-to-one tournament selection based on the fitness values of the candidates (vectors). The better candidate out of the two will have the privilege to move to the next generation.

Each generation of *DE's* search process include these three evolutionary processes (Mutation, Crossover and Selection). The whole process is repeated for *G*(maximum number of generations) number of generations, which is considered as one run in *DE* experiment. The best solution obtained at the end of the run is the solution obtained by *DE* for the given problem, at that particular run. Since all the stages of evolutionary process in *DE* (in fact any *EA*) involves randomness, the average performance of *DE* algorithm in its many runs is used for reporting its performance.

III. Control Parameters of DE

Understanding the influence of the parameters of *DE* (*mutation rate* (*F*), *crossover rate* (C_r) and *population size* (*NP*)) is essential to know the adaptation strategies available for them. This section presents the role of *F*, C_r and *NP*.

The mutation process is to alter the values of the components of each of the candidate in the population. The mutation of *DE* is called as *differential mutation*, since it uses weighted differences of candidates to perform mutation for the current candidate. One among the unique feature of *DE* is its differential mutation. The mutation process can be understood from the Figure 1.

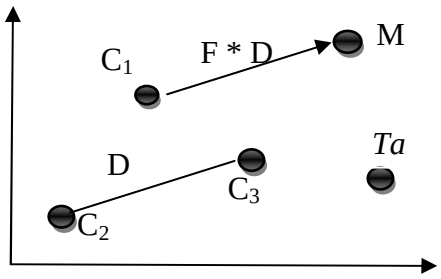

Fig. 1.The Differential Mutation of DE.

The Mutation rate (*F*) (also known as *scaling factor*, *amplification factor*, *mutation step size* or *mutation constant*) is to scale the distance between the pair of vectors C_2 and C_3. This scaled difference is added to the base vector C_1. Thus, the mutation rate is used to control the amplification of the difference vector. Hence, a small value of *F* will lead to premature convergence whereas a larger value will result in a slower convergence. It controls the range of space where the mutant vectors are generated. Thus, it plays an important role in changing diversity in the population.

In classical *DE* algorithm the value for *F* is taken as any real value in the range of 0 to 1. Keeping the value of *F* as constant will deteriorate the diversity of the population during the search, because all the vectors will be created by same difference vector components. So in order to avoid this, many classical *DE* implementations follow a different strategy where *F* will be considered as a random number within the range of [0.2, 0.8]. This ensures that the diversity loss during the search is avoided.

In natural evolution, the crossover process is to create children by inheriting genetic properties from parents. It holds good for *DE* search also. In this process of genetic inheritance, to get diversified candidate from the parent, the parameter crossover rate (C_r) is used. As similar to *F*, C_r also a real valued parameter in the range of 0 to1. It is used to identify the parameters to be inherited from the parents. Crossover rate (C_r) controls the number of elements that the trial vector will inherit from mutant vector and target vector. Thus it defines how different the child vector is from the

parent vectors. In other words, it ensures diversity in the newly created population. Finding the right value of this control parameter is a difficult task as a slight change in the C_r value will affect the efficiency of the algorithm. When C_r value is approximately equals to 0, *DE* makes small explorative moves with higher probability of making improvement. When Cr ≈ 0.9, *DE* makes large explorative moves which helps to perform a more fine-grained search in the solution space and yield large improvements in solution quality.

The third control parameter *NP* also has significant impact on performance of *DE*. If *NP* is small, the search may end in premature convergence, and if it is large the search will take long time to converge. Hence a moderate value for *NP*, to avoid the premature convergence and stagnation, is acceptable for successful *DE* search.

There are many methods proposed for adapting *F* and C_r, many of them were found to be performing better when compared with classical *DE*. However, less works are reported in the literature for adapting *NP*. Hence this paper considers, hereafter, to discuss the existing adaptation strategies for *F* and C_r.

IV. Adaptation Strategies for F and CR

To plan for a suitable adaptation strategy of the control parameters, it is necessary to understand the influence of each of the parameters in the performance of the algorithm.

There are many works reported in literature to discuss the influences of the control parameters. It was in the year 2001, Zaharie [4] was one among the few who started analyzing the possible effect of these control parameter values on *DE* and their critical values. The approach was both a theoretical and empirical study on how the control parameter values are related with population variance of *DE*. An equation to measure the critical values of control parameters was derived. The equation derived by Zaharie was $2F^2p-2p/m + p^2/m+1=0$, where *F* is the scaling factor, *m* is the population size, and *p* is the crossover rate. The value (*F* and *P*) that satisfies this equation was considered critical values. Tremendous efforts have been put by the researcher to analyze the role of each of the *DE* control parameters. Presenting about them is not in the scope of this paper.

Since the impact of the control parameters *MutationRate* (*F*) and *CrossoverRate* (C_r) on the performance of the algorithm is very high, many control parameter adaptation techniques has been put forwarded over the years. All of them have been proved as effective and improving the performance of Differential Evolution algorithm in both converge speed and solution accuracy, compared to the classical *DE*.

The objective of this chapter is to present a brief insight about various adaptation strategies exist in literature. To increase the readability of the paper, the existing adaptation strategies of *F* and C_r are categorized in to four groups with respect to the algorithmic methodology followed by the authors. The details of the categories [57] are

- **Category *1:*** Classical Approach

The strategies in this category use mathematical equations to update the values of the control parameters. This updation is done for every generation or at required time.

- ***Category 2:*** Encoding of control parameters

Another way of adapting control parameter is to encode the control parameters along with the parametric values of the candidates of the population. Hence, the control parameters also evolve as similar to other parameters. The adaptation strategies following this methodology are grouped under this category.

- ***Category 3:*** Deriving from History or Pool

The strategies which use previous information about the performance of the algorithm in the evolutionary search are grouped under this category. The algorithms which maintain pool of values for the control parameters also grouped in this category.

- ***Category 4:*** With added logic

The strategies which use some additional technique or algorithm to adapt the parameters are discussed under this category.

A. Classical Approaches

With the understanding of role of *F* and C_r in the mutation and crossover processes, the works considered in this category uses mathematical equations derived by the authors to update the values of *F* and C_r.

In *SaDE* [5], proposed by A.K Qin and P.N.Suganthan, an adaptive logic for mutation strategy is presented. The mutation strategy is decided based on the success rate calculated for them in the learning period. The *F* and C_r values also calculated differently for each of the mutation strategy. For every individual *i* in the population the *F* and C_r values for the chosen mutation strategy *k* is calculated as follows

$$F_{i,K} = rand(0.5,0.3) \quad (3)$$

$$C_{r\,i,K} = uniform\_rand(C_{r\,mk}, 0.1) \quad (4)$$

where $C_{r\,mk}$ is calculated from a success rule as the mean of C_r. This is followed by many researchers for different applications of *DE* [6].

The *JADE* [7][8][9] proposed by Jingqiao Zhang and Arthur C. Sanderson in the year 2007, introduced a new mutation strategy based on the information obtained about the search progress direction. The values of *F* and C_r are generated newly for each generation with Cauchy and normal distribution. With the initial values of 0.5, the new values at generations are computed as follows.

$$\mu_F = (1-c) * \mu_F + c * Mean_A(S_F) \quad (5)$$

$$\mu_{Cr} = (1-c) * \mu_{Cr} + c * Mean_L(S_{Cr}) \quad (6)$$

where c∈ $(0,1)$, is a constant.

S_F and S_{cr} represent the mean of successful values for crossover rate and scaling factor, respectively. Many other modified *DE* were introduced which borrowed this concept [10]. In the year 2008, Wu Zhi-Feng proposed a new version of *DE* called *AdaptDE* [11]. The fitness values of the trial vector and the target vector are used to find the values for *F* and C_r. Also, the control parameter values for the $G+1^{th}$ generation is calculated by using their values in G^{th}

generation. The fitness values of the trial vector (*ftrv*) and target vector (*ftav*) are compared.

/* F and C_r for i^{th} candidate at $G+1^{th}$ generation*/
If ((*ftrv*<*ftav*) and ($\tau 1 < rand_a$) and ($\tau 2 < rand_b$))

$$F_{i,G+1} = F_{i,G} \quad (7)$$

$$C_{r\,i,G+1} = C_{r\,i,G} \quad (8)$$

Endif
If ((*ftrv*>*ftav*) and ($\tau 1 > rand_a$) and ($\tau 2 > rand_b$)

$$F_{i,G+1} = rand_c \quad (9)$$

$$C_{r\,i,G+1} = rand_d \quad (10)$$

End if

where $rand_a$, $rand_b$, $rand_c$ and $rand_d$ are random numbers within the range [0.1] and $\tau 1$ and $\tau 2$ are probabilities for adjusting F and C_r. The authors preferred a value of 0.1 for both.

In 2009, RadhaThangaraj et al. introduced *ACDE* [12]. In *ACDE*, the whole adaptation process is based on few simple rules. Scaling factor for an individual *i* is defined as

$$F_{G+1} = \begin{cases} F_i + rand_1\sqrt{Grand_1^2 + Grand_2^2}, if P_F < rand_2 \\ F_0 \ otherwise \end{cases} \quad (11)$$

And Crossover rate is found using the rule,

$$C_{r\,G+1} = \begin{cases} C_{r_i} + rand_3, & if P_{CR} < rand_4 \\ C_{r_0} & otherwise \end{cases} \quad (12)$$

where, $Grand_1$ and $Grand_2$ refer to random numbers that are Gaussian distributed which has mean and standard deviation of 0 and 1, respectively. The $rand_1, rand_2, rand_3$ and $rand_4$ are random numbers within the range [0,1]. The P_{Cr} and P_F represent the probabilities for adjusting C_r and F, respectively.

The *IADE* [13] algorithm, introduced by Wenjing Jin et al. in 2010 followed the following adaptation strategy:

- Initially, the values of F and C_r will be fixed as 0.6 and 0.1, respectively.
- It is changed adaptively over the generations.

The mean fitness value of a generation is calculated. To find the F and C_r values for $G+1^{th}$ generation, the mean fitness value of previous generation (G^{th}) is compared with the mean fitness value of $G+1^{th}$ generation, and

If $Meanfitness_{(G)} > Meanfitness_{(G+1)}$ then

$$F_{i,G+1} = F_{i,G} \quad (13)$$

$$C_{r\,i,G+1} = C_{r\,i,G} \quad (14)$$

Else

$$F_{i,G+1} = F_{i,G} + rand\,(0,1) * (F_{max} - F_{min}) \quad (15)$$

$$C_{r\,i,G+1} = C_{r\,i,G} + rand\,(0,1) * (C_{r\,max} - C_{r\,min}) \quad (16)$$

NasimulNoman, in [14], introduced an approach to use the fitness value of the child and the average fitness value of the population to update the control parameters values. It is done as follows, If fitness values of child is less than average fitness value then the control parameter values F and C_r at G^{th} is taken for their $(G+1)^{th}$ generation. Otherwise,

$$F_{i,G+1} = uniform\_rand(0.1,1.0) \quad (17)$$

$$C_{r\,i,(G+1)} = uniform\_rand(0.0, 1.0) \quad (18)$$

In 2012, PengGuo et al. proposed *SelfDE-F* [15]. In *SelfDE-F*, a secondary population is created with the individuals that were discarded during the selection process of the *DE* algorithm. The adaptive method for finding control parameter values for each generation was framed as follows

$$F_{i,G+1} = \begin{cases} w_i * F_{best,G} + (1 - w_i) * rand_1, & if C1 \\ (1 - w_i) * F_{best,G} + w_i * rand_1, & if C2 \\ F_{i,G} & otherwise \end{cases} \quad (19)$$

$$C_{r\,i,G+1} = \begin{cases} C_{r_{min}G} + (CR_{maxG} - CR_{minG}) * w_i & if C3 \\ C_{r\,i,G} otherwise \end{cases} \quad (20)$$

where $w_i = e^{-\frac{(fitness_{best}(G) - fitness_i(G)}{fitness(G)}}$

$$C1 = fitness_i(G+1) < fitness_{i+1}(G)\ and\ rand_2 > \tau_1$$
$$C2 = fitness_i(G+1) < fitness_i(G)\ and\ rand_2 < \tau_1$$
$$C3 = fitness_i(G+1) < fitness_{i+1}(G+1)\ and\ rand_2 > \tau_2$$

$F_{best,G}$ = Scaling factor of the candidate with best fitness value. CR_max_G and CR_min_G are the maximum and minimum crossover rate of generation G. The τ_1 ,τ_2 are two fixed values (Similar to *jDE*) and $rand_1$, $rand_2$ and $rand_3$ are uniform random numbers in the range (0,1).

In the same year, Ali W. Mohammed et al. proposed *ADE* [16]. This paper introduces an alternative differential evolution (*ADE*) algorithm. In *ADE* a new mutation scheme is proposed and control parameters are adapted using a defined equation for both F and C_r. Two values of F are defined, one for the local mutation scheme and the other one for global mutation scheme. Keeping in mind the fact that C_r must start with a small value and must extend to a larger value as the generations increases, the authors framed an equation to C_r as follows,

$$C_r = Cr_{max} + (Cr_{min} - Cr_{max}) * (1 - \frac{G}{GEN})^k \quad (21)$$

where G is the current generation and
GEN is the maximum number of generations.

The authors also mentioned the optimum values for C_{rmin}, Cr_{max} and K as 0.1, 0.8, and 0.4 respectively.

Ali W. Mohammed et al. also proposed *EDE* (Effective Differential Evolution) [17]. A simple method for choosing the values for the control parameters is used. The values for C_r and F are chosen empirically from the range of [0.5, 0.9] and [0.2, 0.8], respectively. This range ensured that there will sufficient exploitation and exploration during the search.

In 2013 *TLBSaDE* was introduced by SubhodipBiswas et al. [18]. *TLBSaDE* borrows the basic concept from *SaDE*. It is based on concept of how learners gain knowledge in class from a teacher. The strategy for adapting control parameters is used as follows: The value of F is taken from a normal distribution with mean and standard deviation of 0.5 and 0.3, respectively. The value of C_r, similar to *SaDE*, is taken from

a normal distribution $N(C_{rm},0.1)$. The value for C_{rm} (mean) is kept as 0.5 and the standard deviation is 0.1.

SAMDE [19] was also introduced in the year 2013 by Xu Wang et al. The *SAMDE* is similar to *JADE*. Scaling factor is found using Cauchy's Distribution with mean μ_F,

$$F_i = randc_i(\mu_F, 0.1) \quad (22)$$

μ_F is initialized to 0.5 and then it is changed as,

$$\mu_F = (1-c) * \mu_F + c * mean_L(S_F), \quad (23)$$

where $mean_L(S_F)$ is the Lehmer mean

Crossover rate is found from a normal distribution of mean μ_{CR} and standard deviation 0.1.

$$CR_i = randn_i(\mu_{CR}, 0.1) \quad (24)$$

$$\mu_{CR} = (1-c) * \mu_{CR} + c * mean_A(S_{CR}) \quad (25)$$

where $mean_A(S_{CR})$ is the arithmetic mean.

Rammohan Mallipeddi and Minho Lee in the same year proposed *ESMDE* [20], an evolving surrogate (substitute) model-based *DE*. In *ESMDE*, based on the current population a surrogate model is created and this is used for selecting appropriate parameter setting so as to creating better off springs during further stages of evolution. Similar to other approaches, the mutation strategies are selected according the concept of pooled values. Here *F* (scaling factor) is selected randomly within the range of [0.5, 1.0] which ensures that there will be adequate exploration alongside with exploitation. Similarly, C_r value is also generated randomly from a range of [0,1].

Quizhen Lin et al. proposed an adaptive algorithm [21], in which C_r is found using a framed equation which ensures that the C_r value will be very large (approximately equal to 0.9) initially and as the generations increases the value decreases and will be stagnant in the range of 0.1 to 0.2. This ensures that the explorative steps taken are very large during the initial phase of the algorithm to favour the global search. As the generations goes the explorative steps will be reduced and the search will be done near local. This ensured good performance of the algorithm. The *ScalingFactor* is found using a Cauchy's Distributions within the range of (0.5, 0.1). The value for *F* is selected from a set of successful values collected for *F*.

Miguel Leon et.al proposed a greedy adaptation of control parameters of *DE* [22]. Here a greedy search will be performed in learning periods that are successive so as to favour continuous and dynamic adjustments of the control parameters *F* and C_r. The whole procedure is as follows, initially the *F* value is set to 0.5 and two neighbours are defined in such a way that a difference d_1 is added and subtracted to the initial *F*, i.e., $(F - d \text{ and } F + d)$. The initial Crossover rate is set to a Cauchy's Distribution with its centre at 0.5 (Cr_m) and scale of 0.2. Two neighbours of C_r are $Cr - d_2$ and $Cr + d_2$. During the predetermined learning period, every candidate and it's neighbouring candidate will have a probability of 1/3 in order to get sufficient number of usages. At the end of the learning period, the best will replace the worst.

Qinqin Fan and Xuefeng Yan proposed another adaptation method [23] to their self-adaptive *DE* strategy (named as *SDE)*. This is an algorithm with zoning evolution of control parameters and adaptive mutation strategies, called as *ZEPDE*. They have defined their own methods to identify the best possible mutation strategies for the *DE* algorithm. The parameter adaptation is done as follows. The total region of *F* and C_r are divided into similar sized four areas. Then at each region the count of control parameter combinations are noted. If the offspring in each zone has a better fitness function value, then it is taken into account that the control parameter combination which gave the best fitness value is the elite one or can be called as Elite Control Parameter Combination (*EPC*). Assume at the h^{th}, the weighted value of each *EPC* is computed as follows,

$$w^G_{N^G_{h,e}} = \frac{f\left(x^{-G+1}_{N^G_{h,e}}\right) - f\left(x^G_{N^G_{h,e}}\right)}{\sum_{N^G_{h,e=1}}^{N^G_{h,elite}} f\left(x^{-G+1}_{N^G_{h,e}}\right) - f\left(x^G_{N^G_{h,e}}\right)} \quad (26)$$

where $N^G_{h,e} = 1,2, \ldots N^G_{h,elite}$

Now the weighted average of control parameters in the h^{th} region is calculated as

$$F^G_{W,h} = \sum_{N^G_{h,e=1}}^{N^G_{h,elite}} \left(w^G_{N^G_{h,e}}\right) * \left(F^G_{N^G_{h,e}}\right) \quad (27)$$

and

$$Cr^G_{W,h} = \sum_{N^G_{h,e=1}}^{N^G_{h,elite}} \left(w^G_{N^G_{h,e}}\right) * \left(Cr^G_{N^G_{h,e}}\right) \quad (28)$$

Once the weighted average is calculated the control parameters for the h^{th} generation is calculated using a cauchy's distribution with mean $F^G_{W,k}$ and $Cr^G_{W,k}$ and standard deviation *(0.55-0.3 *(1-G/Gmax))*.

Xiaowei Zhang and Sanyang Liu proposed APFDE [24] in 2011, drawing inspiration from the theory of electro magnetism. They calculated the charge *Qi* for candidate Xi based on its objective function value and the objective function value of the best candidate in the current generation using the equation given below

$$Q_i = e^{-D} * \frac{f(x_i) - f(x_{best})}{\sum_{i=1}^{NP}(f(x_i) - f(x_{best}))} \quad (29)$$

Later, the equation was modified as shown below:-

$$Q_{ij} = \frac{f(x_i) - f(x_j)}{f(x_{worst}) - f(x_{best})} \quad (30)$$

where *D* is the problem dimension and *NP* is the population size. The Mutant Vector generation in normal *DE* can be written as follows

$$\begin{aligned} V_i &= X_{r1} + F(X_{r2} - X_{r3}) \\ &= X_{r1} + F(X_{r2} - X_{r1}) + F(X_{r1} - X_{r3}) \\ &= X_{r1} + F(X_{r2} - X_{r1}) - F(X_{r3} - X_{r1}) \end{aligned} \quad (31)$$

Using the above derivations, *F* is replaced by Q_{12} and Q_{13} respectively. Taguchichi method along with 2-level orthogonal Array is used to set the value of *Cr*.

Islam et al. [25] proposed *MDE_pBX* that introduced *current-to-g_best* mutation scheme and p-best crossover scheme, along with schemes for updating *F* and C_r in each

generation. The Scale Factor for i^{th} candidate F_i is randomly picked from a Cauchy distribution with location Parameter F_m and scale parameter 0.1. The list of all F values that generated better trial vectors are stored in a set $F_{success}$. F_m is updated using the following equation

$$F_m = F_m * W_f + \left(\sum_{x \in F_{success}} \frac{x^n}{|F_{success}|}\right)^{1/n} * (1 - W_f) \quad (32)$$

Where $W_f = 0.8 + 0.2 * random(0,1)$. C_r is selected randomly from a Gaussian distribution with mean C_{rm} and Standard deviation 0.1. The C_{rm} value is updated in each generation in a similar way as that of F according to the equation given below

$$Cr_m = Cr_m * W_{cr} + \left(\sum_{x \in Cr_{success}} \frac{x^n}{|Cr_{success}|}\right)^{1/n} * (1 - W_{cr}) \quad (33)$$

where, $W_{cr} = 0.9 + 0.1 * random(0,1)$. $Cr_{success}$ is the set of all successful C_r values.

Yet another scheme for adapting F and C_r was proposed in [26], in which the candidates are put into two sorted lists - the first one in descending order based on objective functions and the second one in ascending order based on each candidate's distance from the best candidate. Then the sum of absolute differences between these two ranks for each candidate is calculated, which is called as Indicator of Optimization State (*IOS*). Also IOS_{min} is set to 0 and IOS_{max} is calculated using the following equation

$$IOS_{max} = \begin{cases} \frac{NP^2}{2}, & NP \text{ is even} \\ \frac{(NP+1)*(NP-1)}{2}, & NP \text{ is odd} \end{cases} \quad (34)$$

Then *IOS* is normalized and this normalized value is used to decide whether to explore or exploit. If exploration is to be done, F_{g-1} is increased by 1/10th of ΔF and CR_{g-1} is decreased by 1/10th ΔCR respectively; otherwise F_{g-1} is decreased by 1/10th of ΔF and CR_{g-1} is increased by 1/10th ΔCR respectively. ΔF and ΔCR are computed based on *IOS*, IOS_{max} and IOS_{min} based on the equation

$$\Delta F, \Delta CR = \begin{cases} \frac{IOS - IOS_{min}}{IOS_{max} - IOS_{min}}, & for\ exploration\ step \\ \frac{IOS_{max} - IOS}{IOS_{max} - IOS_{min}}, & for\ exploitation\ step \end{cases} \quad (35)$$

The mathematical equations used in the studies reported in this section are derived / defined by the authors, based on their understanding about the control parameters and their influences in *DE* algorithm. It is worth noting that such equations also adds few new terms to it, which are again to be studied further to set proper value for them. This indirectly increases the complexity of parameter adaptations. To avoid this there exist many parameter adaptation strategies for *DE* control parameters in literature. They allow the values of control parameters also to evolve, as similar to parameters of the candidates, to be better for next generation. Those adaptation strategies encode the control parameters in the parametric representation of the candidates in the population. The next section discusses the strategies with such encoding scheme.

B. Control Parameter Encoding

To avoid inclusion of additional parameters in adapting the required parameters, the idea of encoding the control parameters with the individual candidates in the population arose. This encoding let the control parameters also to evolve along with other parts of the candidates. At the required stages the F and C_r values are selected suitably from the evolved values of them. Hence, these strategies include efficient selection mechanism to consider the evolved values of F and C_r available at the parent candidates of mutation and crossover. The works similar to this strategy are discussed in this chapter.

Mahamed G. H. Omran et al. introduced a *self-adaptiveDE* [27] in 2005. This algorithm uses the differential mutation mechanism to find the new values of F and C_r. Each individual i in the population is encoded with F_i and C_{ri}, and these values are calculated as follows

$$F_i(t) = F_{i_1}(t) + N(0,1) * (F_{i_2}(t) - F_{i_3}(t)) \quad (36)$$

where i_1, i_2 and i_3 are random and distinct candidates chosen with a uniform distribution $U(1,\ldots,NP)$

Another self adaptive *DE* named *SADE_ALM* (Self Adaptive Differential Evolution with Augmented Lagrange Multiplier) [28][29] was proposed by C. Thitithamrongchai and B. Eua-Arporn in 2006. In the *SADE_ALM* the F and C_r are encoded in the first two positions of the candidates. The F and C_r values are initialized as follows

$$F_i = F_{i,low} + \rho_{1i} * (F_{i,hi} - F_{i,low}) \quad (37)$$

$$CR_i = CR_{i,low} + \rho_{1i} * (CR_{i,hi} - CR_{i,low}) \quad (38)$$

Then the F and C_r values are undergoing the mutation and crossover operations. The mutation process is done as follows:

$$F_i^{|} = F_{i,r3} + F_i * (F_{i,r1} - F_{i,r2}) \quad (39)$$

$$CR_i^{|} = CR_{i,r3} + CR_i * (CR_{i,r1} - CR_{i,r2}) \quad (40)$$

where r_1, r_2 and r_3 are non-equal indices between 1 and *NP*. The crossover process is done as follows:

$$F_i^{||} = \begin{cases} F_i^{|}, & if\ \forall \rho_{1i} < CR_i or\ j = j_{rand} \\ F_i & otherwise \end{cases} \quad (41)$$

$$CR_i^{||} = \begin{cases} CR_i^{|}, & if\ \forall \rho_{1i} < CR_i or\ j = j_{rand} \\ CR_i & otherwise \end{cases} \quad (42)$$

Finally, at required points, the F and C_r values are chosen from the best candidate of the population. Amin Nobakhti and Hong Wang,in 2006, proposed an adaptive *DE* [30][31] with control parameter adaptation, mainly for the mutation rate (F). Each population vector is assigned with their own value of F, which is initialized by a uniform distribution bounded by F_l and F_{up}.During the process of evolution, once the trial vector has been created and is found to be better than the target, the trial vector will inherit the value of F from the target. After every fixed 'k' generations, the entire population is analyzed for accumulated improvements and is sorted accordingly. From this sorted population, a threshold value is identified which represents the accumulated

improvements of at least half the population. For all the individuals, whose F_i value is greater than this threshold, F values are retained and for others new F_i values are randomly generated

In the year 2007, J.Brest et al. put forwarded the *jDE* algorithm [27][32][33][34][35]. In *jDE* also, the control parameters F and C_r are encoded with the population along with its genes. These values are adapted during every generation according to two other fixed values 'τ_1' and 'τ_2'. The value of F in the $(G+1)^{th}$ generation will be same as the value in G^{th} generation if a randomly created number within the range (0,1) is greater than τ_1, else F value is found as

$$F_{i,G+1} = F_{l,G} + rand * (F_{u,G}) \quad (43)$$

Suffix '*l*' and '*u*' stands for lower and upper values of F. Similarly, for C_r if 'τ_2' is less than randomly generated value then the previous generation value is retained, else

$$CR_{i,G+1} = rand(0,1) \quad (44)$$

An enhanced version *jDE*, named *jDE-2* [36], was introduced later with an added concept for keeping the bound-constraints problem feasible. But the parameter control part was kept same as that of *jDE*.

Brest also introduced a different variant of the above called *jDEdynNP-F* [37] in 2008, where along with control parameter adaptation, population size reduction mechanism is also implemented. The F and C_r adaptation of the proposed algorithm remain the same as that of its previous version. Similar adaptation mechanism for F and C_r is followed by Zhong-bo Hu et al. [38].

Also in the same year Chukiat Worasucheep, proposed *wDE* [39]. In *wDE*, a separate strategy adaptation and parameter adaptation is introduced. In parameter adaptation, each individual '*i*' is extended with corresponding C_{ri} and F_i which are uniformly initialised within the range [0,1] and [0,2], respectively, at the beginning. A fixed number of generation period (learning period) is considered and adaptation of these control parameters happen after these fixed number of generations. Also two variable *nsi* and *nfi* are introduced which indicates the number of success and failure for a particular individual *i* for entering into the next generation with respect to the learning period. The probability of pass (*PPi*) for a particular individual *i* and the average of pass probabilities (*PPavg*) are calculated. Now, for all those individuals, whose probability of pass is below the probability pass average, the control parameters are updated as follows

$$F_i = uniform\_rand(0,2) \quad (45)$$
$$CR_i = uniform\_rand(0,1) \quad (46)$$

AlesZamuda and BorkoBoskovic in 2007 came up wit*DEwSAcc* [40], Differential Evolution With Self-adaptation and Cooperative Co-evolution. In *DEwSAcc* also, as similar to other works in this category, all the individuals in the population are extended to include their own control parameters, F_i and C_{ri}. The control parameter value of the next generation *(G+1)* depends on the values of these control parameters of the current generation (*G*). F and C_r are updated as follows

$$F_{i,G+1} = F_{i,G}e^{\tau N(0,1)} \quad (47)$$
$$C_{r\,i,G+1} = C_{r\,i,G}e^{\tau N(0,1)} \quad (48)$$

where, τ represents the learning period which is $1/\sqrt{D}$.

In 2008, Omar S. Soliman and Lam T. Bui came up with a self-adaptive strategy to use Cauchy distribution [41]. The control parameters for each individual are encoded along with them. For an individual *i*, scaling factor is found by

$$F_{i,t+1} = \begin{cases} C(\mu, \delta_{i,t+1}), & if rand_1 \leq \pi_1 \\ C(\mu, \delta_{i,t+1}) & otherwise \end{cases} \quad (49)$$

where, $\delta_{i,t+1} = \delta_l + \delta_u * rand_2$ and $\delta_l$and δ_u are the lower and upper limit of possible values for scaling factor.
Crossover rate is found as follows:

$$C_{r\,i,t+1} = \begin{cases} rand_3, if rand_4 \leq \pi_2 \\ C_{r\,i,t'} & otherwise \end{cases} \quad (50)$$

$rand_k \in [0,1]$, k = 0, 1,2,3,4 are uniform random numbers. The probability to adapt F and C_r are denoted by π_1 and π_2.

Grant Dick [42] in 2010 defined *SaNSDE* (self-adaptive neighbourhood search differential evolution) which uses the neighbourhood search which is one of the core concepts of Evolutionary Computing. Control Parameters are added to each individual. It works as follows, the first step is to initialise the CR_m value to 0.5 .Then similar to above it is found during each generation using a normal distribution with mean value CR_m and standard deviation 0.1 for each of the individual. F value is found as follows,

$$F_i = \begin{cases} N_i(0.5,0.3), if\ U_i(0,1) < f_p \\ C_i(0.0,1.0), & otherwise \end{cases} \quad (51)$$

where $U_i(0,1)$ is a uniform random number in range [0,1).

The SaFDE [43] proposed by Teng NgaSing et al. encodes scale factor inside each candidate. Initial population has F randomly initialized for each candidate. During trial vector generation, if the random number generated is less than cross over probability, trial vector's scale factor is also updated in the same way as the other genes using the differential mutation as given below

$$TrialVector.F = X1.F + \alpha_{DE} * (X2.F - X3.F) \quad (52)$$

Here α_{DE} is a randomly chosen value between 0 and 1. If the calculated value of F goes beyond the limits [0,1], then it is randomly re-initialized. Authors have self adapted only F and *Cr* and they are selected randomly from [0.1, 1.0].

In the year 2013, Ming Yang et al. proposed a variation of *jDE* called *PA-jDE* [44] that does a population adaptation along with adaptation of F and C_r. F and C_r is encoded with

each individual. F_i and C_{ri} are updated in each generation based on two thresholds *T1* and *T2* as follows:

$$F_{i,G+1} = \begin{cases} F_l + r_1 * F_u, r_F < T1 \\ F_{i,G}, otherwise \end{cases} \quad (53)$$

$$CR_{i,G+1} = \begin{cases} r_2, r_{Cr} < T2 \\ Cr_{i,G}, otherwise \end{cases} \quad (54)$$

where r_1, r_2, r_F, and r_{Ct} are random numbers. F_l is set to 0.1 and F_u is set to 0.9. *T1* and *T2* were set to 0.1.

The above discussed adaptation strategies of F and C_r are proven to be working better than classical adaptation strategies. They add only little complexity to the algorithm, because it does the changes only in the parametric representation.

C. Deriving from History or Pool

It is also interesting that the future values for the control parameters of any algorithm is decided based on the performance of the algorithm with the past values of the control parameters. There are number of research works reported in *DE* literature too, in this direction. As well as, all the possible values for each of the control parameters are pooled and the algorithm is allowed to choose the required values based on the present performance of the algorithm. All such works are considered for discussion in this section.

This section is to discuss the adaptation strategies deriving values for the control parameter from the performance history of the algorithm or from corresponding pool of values.

The adaptive *DE* proposed by Hui-rongetal [45], used previous learning experiences to choose the values for the control parameters. The values of F and C_r are found at each generation. A random number (r_n) with uniform distribution is generated and

If $r_n < 0.2$

$$F_{i,G+1} = F_l + rand * (F_u - F_l) \quad (42)$$

$$CR_{i,G+1} = CR_l + rand * (CR_u - CR_l) \quad (55)$$

Else

$$F_{i,G+1} = F_{i,G} \quad (44)$$

$$CR_{i,G+1} = CR_{i,G} \quad (56)$$

End if

In *SHADE* proposed by Ryoji Tanabe and Alex Fukunaga [46], the history information about the successful parameter values are maintained to guide *DE* search. It has a memory to store H values of F and C_r. Then the values for F_i and C_{ri} are selected form the range [1,H] with a random index *ri*.

Another success history based model known as *DEsPA* was proposed by Noor Awad et al, in 2015 [47]. Along with F and C_r the *NP* value also adapted in *DEsPA*. Every individual i in the population is assigned with its own greediness factor p_i. A 2-D memory structure stored with the mean values of F (μ_F) and Cr (μ_{Cr}) is used for control parameter adaptation. The size of memory (M) is set as half of *NP*. A random index $r_i \in (0,1)$ is chosen and is used for find F and C_r.

$$F_i = randc_i(\mu F_{ri}, 0.1) \quad (57)$$

$$C_{ri} = randn_i(\mu C_{rri}, 0.1) \quad (58)$$

The *EPSDE* proposed by R Mallipeddi et al used the concept of pooled values [48]. It consists of a pool of mutation strategies and pools for corresponding parameters for those strategies. The F and C_r values are taken from a pool which has values within the range [0.1, 0.9] and [0.4, 0.9], respectively. Every step changes the values by 0.1. In *EPSDE*, each individual in the population is associated with a random mutation strategy taken from the pool. Along with the matched mutation strategy, the corresponding F and C_r values are also chosen. These values and strategies will survive until the target vector performs poorer when compared with the trial vector. Once this condition is failed, a new mutation strategy will be associated with the target vector. These strategies could be selected from the pool or from the successful combination stored before.

The *CoDE* [49] algorithm was also introduced in 2011 by Y Wang et al. The parameter values are predefined here. Based on the carefully selected three mutation strategies, the parameter values will be changed. During each generation a set of three trial vectors are generated. On comparison with the target vector, the best of the four (3 trial and 1 target) will go to the next generation. Three combinations are made for F and C_r based on the mutation strategies. It will follow either of the three defined as [F=1.0, C_r=0.1], [F=1.0, C_r =0.9] and [F = 0.8, Cr = 0.2].

Wenyin Gong et al. presented a variant of *JADE*, called R_{cr}-*JADE* in [50], that repairs C_r based on Success history of C_r values. C_{ri} and F_i are selected from Normal Distribution $N(\mu_{cr}, 0.1)$ and Cauchy Distribution $C(\mu_F, 0.1)$respectively for i^{th} candidate. Then CR_i is repaired as per the equation given below

$$Cr_i = \frac{m}{D} \quad (59)$$

where m is the number of genes that were copied from mutant vector and D is the problem dimension. If trial vector U_i is better than target vector X_i, then Cr_i and F_i are added to lists of successful Crossover and Mutation Parameters (S_{Cr} and S_F). Then μ_{cr} is updated as the arithmetic mean of all the C_r values in S_{Cr}. Similarly μ_F is updated as the Lehmer mean of all the F values in S_F.

Comparing to other three categories, this category covers very less works in the literature. This is because of the complexity involved in remembering required information from sufficient past time and choosing the values to be achieved in the pool.

D. Parameter Adaptation with Added Logic

Another commonly used strategy for *DE* control parameter adaptation is to insert an additional component (or algorithm or logic) to the structure of *DE*. This added component will monitor *DE's* performance in solving the given problem and suitably adapt the required parameters. The researchers have used some other existing algorithm as the component or have designed their own algorithm. The former one is termed in other words as hybridization of *DE* with other algorithms.

An algorithm hybridizing *DE* and Fuzzy Logic, named as *FADE* (Fuzzy Adaptive Differential Evolution) [51], was proposed by Lampinen and Liu in the year 2002. In *FADE*, a

fuzzy logic controller was used to find the values for F and C_r for the candidate i (F_i and C_{ri}). The FLC-MODE (Fuzzy Logic Controlled Multi-objective Differential Evolution) [52] was introduced by FengXue et al in the year 2005, as similar to *FADE*.

An Iterative Function System Based Adaptive *DE* was proposed by Ya-Liang Li, et. al. [53]. In this algorithm the control parameters are adapted using an iterative function system. The $F_{i,G}$ and $CR_{i,G}$ values are adapted using the following equations

$$F_{i,G+1} = \begin{cases} \alpha_1 F_{i,G} + (1-\alpha_1) r_1, & iff(u_{i,G}) < f(x_{i,G}) \\ \alpha_2 F_{i,G} + (1-\alpha_2) r_2 & otherwise \end{cases} \quad (60)$$

$$CR_{i,G+1} = \begin{cases} \alpha_3 CR_{i,G} + (1-\alpha_3) r_1, & iff(u_{i,G}) < f(x_{i,G}) \\ \alpha_4 F_{i,G} + (1-\alpha_4) r_2 & otherwise \end{cases} \quad (61)$$

Here, $\alpha_1, \alpha_2, \alpha_3$ and α_4 are uniformly generated within the range $(0,1)$. The parameter r_1 is set to 0.5 and r_2 is set to 1.

Patricia Ochoa et.al proposed *FDE* (Fuzzy Differential Evolution), which uses concept of fuzzy system for parameter adaptation [54]. The fuzzy system added to *DE* will give the best possible values for the control parameters. The fuzzy system has 3 membership functions (FN_1, FN_2 and FN_3) to mean the low, medium and high values of the parameters. It also used 3 fuzzy rules to update the values of the control parameters.

In 2009, M.G. Epitropakis et al. introduced an evolutionary approach towards self-adapting *DE*, known as *ESADE* [55]. In *ESADE*, a unique strategy was followed in finding the values of the control parameters. It uses two *DE* algorithms, one is to find the mutation rate (F), and the other for optimizing the given objective function. In the first *DE* algorithm to find F value, a one-dimensional population is initialized as follows,

$$Xg = \{Fg\} \quad (62)$$

where Fg corresponds to possible values of F. Rather than initializing it with values in the range (0.1, 1.0], based on their study they have initialized the population with values from a normal distribution with mean 0.5 and standard deviation 0.3. Once, the population has been initialized in the first *DE*, one generation of the second algorithm is performed. Here the fitness value of the best candidate ($f(x_{gbest})$) is taken and it is considered as the fitness value of corresponding individual of the first algorithm. For adapting C_r, a normal distribution with mean 0.6 and standard deviation 0.1 is considered, and values are taken from this normal distribution at every generation. Thus, in *EPSADE*, the first algorithm gives the Scaling factor value and using this value the second *DE* algorithm optimizes the given objective function.

Pravakar Roy et.al proposed Differential Evolution that is Genetically Programmed [56] which ensures a self-adaptive mechanism in the *DE* algorithm. Here, the initial preparations are made in such a way that the need of F is null. The system finds out the best crossover rate as follows, for each individual in the population of *GP*, a C_r value is also associated with it and it is updated during the natural evolution process of *GP*. Initially it is taken from a Gaussian distribution and later the *GP* will alter the values based on the predetermined fitness value. Also a counter is kept for the number of times the alteration has performed.

The adaptation strategies discussed in this section have used additional components to tune the values for the control parameters.

V. F and Cr Adaptation Strategies - Insight

The research works focusing on control parameter adaptation of *DE* algorithm are grouped in to four categories and presented in Table 1. Due to large number of reports available in the literature, this paper aimed to consider the research works for adapting the parameters F and C_r.

TABLE I
List of papers under each category

Categories			
I Classical Approaches	**II Encoding of Parameters**	**III Deriving from History/Pool**	**IV With Added Logic**
[5],[6],[7], [8],[9],[10], [11],[12],[13], [14],[15],[16], [17],[18],[19],[20],[21],[22],[23],[24],[25], [26].	[27],[28],[29], [30],[31],[32], [33],[34],[35], [36],[37],[38],[39],[40],[41],[42], [43],[44].	[45],[46],[47] [48],[49],[50]	[51],[52], [53],[54], [55],[56]

The research works in Category I use author defined equations to calculate the values for the parameters. Many authors also have considered using statistical distribution to select the values for the control parameters. In category II, the algorithms which encode the control parameters along with other parameters of the candidates are considered. Evolution of those parameters is done by normal *DE* process or by some other newly added algorithm. Recording the history of behaviour of *DE* in previous generations and deriving necessary information from them to decide the control parameter values for the forthcoming generations is another strategy for parameter adaptation. Research works using this strategy are grouped under this category III. Finally, in Category IV the works which consider to add additional component to *DE* for parameter adaptation are grouped.

VI. Conclusions

The critical parameters of *DE* algorithm are F, C_r and NP. Selecting suitable values for them are very important as well as crucial for successful application of *DE* for any optimization problem. There exists no standard method for choosing values for these parameters. However, to alleviate this many parameter adaptation strategies are proposed in the literature. The existing adaptation strategies are identified and are categorized in to four groups, and brief insight about each of the identified strategies are presented in this paper. The categories of adaptation strategies presented in this paper are strategies with classical approaches, strategies with

encoding of parameters, strategies using history/pool and strategies adding new components.

REFERENCES

[1] Storn, R. and Price, K. (1995), "Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report-95-012, ICSI.

[2] K. V. Price, R .M.Storn, and J.A.Lampinen (2005), "Differential Evolution –A practical approach towards global optimization", 1st ed. New York: Springer-Verlag, Dec.2005.

[3] Storn, R. and Price, K. (1997), "Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces", Journel of Global Optimization 11, pp.341-359.

[4] Zaharie, D. (2001), "On the explorative power of differential evolution algorithms", Proceeding of the 3rd International Workshop on Symbolic and Numeric Algorithms on Scientific Computing, SYNASC-2001.

[5] A.K.Qin and P.N.Sugathan (2005), "Self-Adaptive Differential Evolution Algorithm for Numerical Optimization", IEEE Congress on Evolutionary Computation, CEC 2008, pp. 3718-3725.

[6] Xuexia Zhang, Weirong Chen, Chaohua Dai, Ai Guo(2008), "Self-adaptive Differential Evolution Algorithm for Reactive Power Optimization", Fourth International Conference on Natural Computation, pp. 560-564.

[7] Jingqiao Zhang and Arthur C. Sanderson (2007), "JADE: Adaptive Differential Evolution With Optional External Archive", IEEE transaction on Evolutionary Computation, pp. 945-958.

[8] Jingqiao Zhang and Arthur C. Sanderson (2007), " JADE: Self-Adaptive Differential Evolution with Fast and Reliable Convergence Performance", IEEE Congress on Evolutionary Computation, pp. 2251-2258.

[9] Ales Zamuda, Janez Brest, Borko Boskovic (2007), "Differential Evolution for Multi-objective Optimization with Self Adaptation", IEEE Congress on Evolutionary Computation, pp. 3617-3624.

[10] Jingqiao Zhang and Arthur C. Sanderson (2008), "Self-Adaptive Multi-Objective Differential Evolution with Direction Information Provided by Archived Inferior Solutions", 2008 IEEE Congress on Evolutionary Computation (CEC 2008), pp. 2801-2810.

[11] Wu Zhi-Feng, Huang Hou-Kuan, Yang Bei (2008), "A Modified Differential Evolution Algorithm with Self-Adaptive Control Parameters", Proceedings of 2008 3rd International Conference on Intelligent System and Knowledge Engineering, pp. 524-527.

[12] Radha Thangaraj, Millie Pant and Ajith Abraham (2009), "A Simple Adaptive Differential Evolution Algorithm", 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 457-462.

[13] Wenjing Jin , Liqun Gao , Yanfeng Ge , Yang Zhang. (2010), "An Improved Self-adapting Differential Evolution Algorithm", 2010 International Conference on Computer Design and Application (ICCDA), pp. V3-341-344.

[14] Nasimul Noman, Danushka Bollegala and Hitoshi Iba (2011), "An Adaptive Differential Evolution Algorithm", IEEE Congress on Evolutionary Computation-2011, pp. 2229-2236.

[15] Peng Guo, Naixiang Li, Tonghai Liu(2011), "Parameter Self-Adaptive Differential Evolution Algorithm with Secondary Population", 2013 Ninth International Conference on Computational Intelligence and Security, pp. 81-85.

[16] Ali W. Mohamed, Hegazy Z. Sabr, Motaz Khorshid,(2012) "An alternative differential evolution algorithm for global optimization", Journal of Advanced Research,2012, pp.149-65.

[17] Ali Wagdy Mohammed , Hegazy Zaher Sabry , Tareq Abd-Elaziz(2012), "Real Parameter optimization by an effective Differential evolution algorithm", Egyptian Informatics Journal,2013, pp. 37-53.

[18] Subhodip Biswas, Souvik Kundu1, Swagatam Das and Athanasios V. Vasilakos (2013), "Teaching and Learning best Differential Evolution with Self Adaptation for Real Parameter Optimization", 2013 IEEE Congress on Evolutionary Computation, pp. 1115-1122.

[19] Xu Wang, Shuguang Zhao, Yanling Jin, Lijuan Zhang(2013), "Differential Evolution Algorithm based on Self-adaptive Adjustment Mechanism [SAMDE]", 2013 25th Chinese Control and Decision Conference (CCDC), pp. 577-581.

[20] Rammohan Mallipeddi , Minho Lee (2015), "An Evolving Surrogate model-based Differential Evolution algorithm", Applied Soft Computing 34 (2015), pp. 770–787.

[21] Quizhen Lin, Qingling Zhu, Peizhi Huang, Jianyong (2015), "A novel hybrid multi-objective immune algorithm with adaptive Differential evolution", Computers and Operations Research 62 (2015), pp. 95-111.

[22] Miguel Leon ,Ning Xiong (2013) "Greedy Adaptation of Control Parameters in Differential Evolution for Global Optimization Problems", 2013 IEEE Symposium on Differential Evolution pp. 38-45.

[23] Qinqin Fan and Xuefeng Yan, (2015) "Self Adaptive Differential Evolution with Zoning Evolution of Control Parameters and Adaptive Mutation Strategies", 2015 IEEE Transactions on Cybernetics.

[24] Xiaowei Zhang, Sanyang Liu, (2011) " Almost-Parameter-Free-DE", IEEE Seventh International Conference on Natural Computing, 2011, pp. 1461-1465.

[25] Sk.Minhazul Islam, Swagatam Das, Saurav Ghosh, Subhrajit Roy, P.N.Suganthan (2012), "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization", IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics, VOL. 42, NO. 2, APRIL 2012, pp. 482-500.

[26] Wei-jie Yu, Jun Zhang, (2012) "Adaptive Differential Evolution with Optimization State Estimation", GECCO'12, July 7–11, 2012, Philadelphia, Pennsylvania, USA, pp. 1285-1291.

[27] J Brest , M Mernik(2007), "Population size reduction for the Differential Evolution Algorithm.", Springer Applied Intelligence, Volume 29 , pp. 228-247.

[28] C. Thitithamrongchai, and B. Eua-Arporn (2006), "Hybrid Self-adaptive Differential Evolution Method with Augmented Lagrange Multiplier for Power Economic Dispatch of Units with Valve-Point Effects and Multiple Fuels", Power Systems Conference and Exposition-PSCE, pp. 908-91.

[29] C.Thitithamrongchai and B.Eua-Arpon (2006), "Economic Load Dispatch for Piecewise Quadratic Cost Function using Hybrid Self-Adaptive Differential Evolution Algorithm with Augmented Lagrange Multiplier Method ", International Conference on Power System Technology, pp. 1-8.

[30] Amin Nobakhti and Hong Wang (2006), "Co-evolutionary Self-Adaptive Differential Evolution with a Uniform-distribution Update Rule", IEEE International Symposium on Intelligent Control, October 2006, pp. 1264-1269.

[31] Amin Nobakhti and Hong Wang (2006), "A Self-adaptive Differential Evolution with application on the ALSTOM gasifier", Proceedings of the 2006 American Control Conference,Minneapolis, Minnesota, USA, June 14-16, 2006, pp. 4490-4494.

[32] Janez Brest, Saso Griener, Borko Boskovic, Marjan Mernik (2006), "Self-Adapting Control Parameters in Differential Evolution: A comparative study on numerical benchmark functions", IEEE Transactions on Evolutionary Computations,, VOL. 10, NO. 6, December 2006, pp. 646-657.

[33] Janes Brest, Viljem Zumer, Mirjam Sepsy Maucec (2006) , "Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization", IEEE Congress on Evolutionary Computations, Vancouver, July 2006, pp. 215-222.

[34] Janez Brest, Aleˇs Zamuda, Borko Boˇskovi´c, Mirjam Sepesy Mauˇcec, and Viljem ˇZumer (2009), "Dynamic Optimization using Self-Adaptive Differential Evolution", 2009 IEEE Congress on Evolutionary Computation (CEC 2009), pp. 415-422.

[35] Rupam Kundu,Rohan Mukharjee,Swagatam Das,Athanasios V, (2013) "Adaptive Differential Evolution with Difference Mean Based Perturbation for Dynamic Economic Dispatch Problem", 2013 IEEE Symposium on Differential Evolution pp. 38-45.

[36] J.Brest, B.Boskovic,V.Z.S. Greiner, and M.Maucec.(2007) "Performance Comparison of self-adaptive and adaptive differential evolution algorithms", Soft Computing- A Fusion of Foundations, Methodologies and applications, 11(7):617-629.

[37] Janez Brest, Aleˇs Zamuda, Borko Boˇskovi´c, Mirjam Sepesy Mauˇcec, and Viljem ˇZumer (2008), "High-Dimensional Real-Parameter Optimization using Self-Adaptive Differential Evolution Algorithm with Population Size Reduction", 2008 IEEE Congress on Evolutionary Computation (CEC 2008), pp. 2032-2038.

[38] Zhong-bo Hu, Qing-hua Su, Sheng-wu Xiong, Fu-gao Hu (2008), "Self-adaptive Hybrid Differential Evolution with Simulated Annealing Algorithm for Numerical Optimization", 2008 IEEE Congress on Evolutionary Computation (CEC 2008), pp. 1189-1194.

[39] Chukiat Worasucheep (2007), "A new Self Adaptive Differential Evolution: Its Application in Forecasting the index of Stock

Exchange in Thailand", IEEE Congress on Evolutionary Computation, pp. 1918-1925.
[40] Ales Zamuda, Janez Brest, Borko Boskovic, Viljem Zumer (2008), "Large Scale Global Optimization Using Differential Evolution with Self-adaptation and Cooperation Co-evolution", IEEE Congress on Evolutionary Computation, pp. 2251-2258.
[41] Omar S. Soliman and Lam T. Bui (2008), "A Self-Adaptive Strategy for Controlling Parameters in Differential Evolution", 2008 IEEE Congress on Evolutionary Computation (CEC 2008), pp. 2837-2842.
[42] Grant Dick, (2010) "The utility of Scaling Factor Adaptation in Differential Evolution". Grant Dick," WCCI 2010 IEEE World Congress on Computational Intelligence, July, 2010, pp. 4355-4362.
[43] Teng Nga Sing, Jason Teo, Mohd.Hanfi Ahmad Hijazi, (2007) "Self-adaptive Scling Factor in Differential Evolution", Regional Conference on Computational Science and Technology , School of Engineering and Information Technology, Universiti Malaysia Sabah,29-30 November 2007, pp.112-116.
[44] Ming Yang, Zhihua Cai,Changhe Li, (2013) "An Improved Adaptive Differential Evolution Algorithm with Population Adaptation", GECCO'13, Amsterdam, The Netherlands , July 6–10, 2013, pp.145-152.
[45] Li Hui-rong , Gao Yue-lin , Li Chao ,Zhao Peng-jun (2011), "Improved Differential Evolution algorithm with adaptive Mutation and control parameters", 2013 Ninth International Conference on Computational Intelligence and Security, pp. 81-85.
[46] Ryoji Tanabe and Alex Fukunaga (2013), "Success-History Based Parameter Adaptation for Differential Evolution(SHADE)", Evolutionary Computation (CEC) 2013 IEEE Congress, pp. 71-78.
[47] Noor Awad, Mostafa Z.Ali, Robert G.Reynolds (2015), "Differential Evolution Algorithm with Success based Parameter Adaptation for CEC2015 Learning based Optimization", IEEE Congress on Evolutionary Computation 2015, pp. 1098-1105.
[48] R Mallipeddi, S Mallipeddi, P.Sugathan(2010), "Differential Evolution with ensemble of parameters and mutation strategies", Elsevier, Applied Soft Computing 11, pp. 1679-1696.
[49] Y.Wang , Z.Cai, Q.Zhang (2011), "Differential Evolution wiith composite trial vector generation strategies and control parameters", IEEE Transactions on Evolutionary Computation Vol.15, pp. 55-66.
[50] Wenyin Gong, Zhihua Cai, Yang Wang, (2014) "Repairing the Crossover Rate in Adaptive Differential Evolution", Elsevier Applied Soft Computing, Vol.15, April 2014, pp. 149-168.
[51] Junhong Liu and Jouni Lampinen (2002) , "A Fuzzy Adaptive Differential Evolution Algorithm", IEEE TENCON , pp. 606-611.
[52] Feng Xue , Arthur C.Sanderson , Piero P. Bonissone ,Robert J. Graves (2005), " Fuzzy Logic Controlled Multi-Objective Differential Evolution", IEEE International Conference on Fuzzy Systems, pp. 720-725.
[53] Ya-Liang Li, Fei Ding, and Yu-Xuan Wang (2008), "Iterated Function System Based Adaptive Differential Evolution Algorithm", IEEE Congress on Evolutionary Computation, pp. 1290-1294.
[54] Patricia Ochoa, Oscar Castillo, Jose SoriaFukunaga (2013), "A Fuzzy Differential evolution method with dynamic adaptation of parameters for the optimization of Fuzzy Controllers", Evolutionary Computation (CEC) 2013 IEEE Congress, pp 1-6.
[55] M.G. Epitropakis, V.P. Plagianakos, and M.N. Vrahatis (2009), "Evolutionary Adaptation of the Differential Evolution Control Parameters", 2009 IEEE Congress on Evolutionary Computation (CEC '09), pp. 1359-1366.
[56] Pravakar Roy, Md. Jahidul Islam and Md. Monirul Islam, (2012) "Self Adaptive Genetically programmed Differential Evolution", 2012 7th International Conference on Electrical and Computer Engineering pp. 639-642.
[57] P. Pranav, and G. Jeyakumar, (2015) "Control Parameter Adaptation Strategies for Mutation and Crossover Rates of Differential Evolution Algorithm – An Insight," In Proceedings of IEEE International Conference on Computational Intelligence and Computing Research (ICCIC-2015), December-2015, pp. 563 - 568.