Evolving the Parameters of Differential Evolution Using Evolutionary Algorithms

Saber Elsayed and Ruhul Sarker

School of Engineering and Information Technology, University of New South Wales at Canberra (UNSW Canberra), Australia
{s.elsayed,r.sarker}@adfa.edu.au

Abstract. Differential evolution has shown tremendous success in solving different complex optimization problems. However, the performance is highly dependent on the selection of its parameters. Although many techniques have been introduced to adaptively (or self-adaptively) determine the parameters, the task is recognized as a tedious one. In this research, we investigate the use of evolutionary algorithms, such as covariance adaptation matrix evolution strategy, differential evolution and genetic algorithm, to self-adaptively determine the possible values of both the amplification factor and crossover rate. The performances of the algorithms are compared to each other, as well as to a standard differential algorithm, by solving a well-known set of benchmark problems. The experimental results show that such an approach can improve the performance of differential evolution, however further investigation is required to find the appropriate evolutionary algorithm for evolving parameters.

Keywords: differential evolution, covariance adaptation matrix evolution strategy, genetic algorithm, self-adaptation.

1 Introduction

The evolutionary algorithms (EAs), such as genetic algorithms (GA) [1], differential evolution (DE) [2] and evolution strategies (ES) [3], are popular choice to many researchers and practitioners for solving their complex optimization problems. Among EAs, DE has shown its superiority to many other algorithms in solving problems with different mathematical properties. However, it is well-known that DE parameters such as amplification factor (F), crossover rate (Cr) and population size (PS) play a vital role on its success, which led researchers to investigate this research topic, and propose different adaptive and self-adaptive mechanisms to avoid a trial-and-error approach in the selection of parameters. This directs us to the no-free launch theorem [4], which shows that one set of parameters may be well suited for a set of problems that may not work well for another problem, or another class, or range of problems.

While solving an optimization problem, one of the interesting mechanisms to self-adaptively determine the DE parameter values is using an evolution process that may involve DE or any other EAs. This mechanisms dates back to 2002 when Abbass [5] proposed a self-adaptive operator (crossover and mutation) for multi-objective

H. Handa et al. (eds.), *Proc. of the 18th Asia Pacific Symp. on Intell. & Evol. Systems – Vol. 1*,
Proceedings in Adaptation, Learning and Optimization 1, DOI: 10.1007/978-3-319-13359-1_40

optimization problems, where F was generated using a Gaussian distribution $\mathrm{N}(0,1)$ and then updated using a DE algorithm. This technique has been modified in [6-9], in which DE variants with more than one difference vectors (DV) were used to evolve parameters, as well as F and Cr were initialized using $\mathrm{N}(0.5,15)$. However, to the best of our knowledge, the use of other EAs to self-adaptively determine DE parameters is rare.

Other mechanisms, which do not depend on EAs, have also been proposed. Qin *et al.* [10] proposed a DE algorithm (SaDE). In it, F was approximated by a normal distribution N (0.5, 0.3), and truncated to the interval (0, 2]. The crossover probabilities were randomly generated according to an independent normal distribution with mean $\mathrm{Cr_m}$ and a standard deviation value of 0.1. The $\mathrm{Cr_m}$ values remained fixed for five generations before the next re-generation. $\mathrm{Cr_m}$ was initialized to 0.5, and it was updated every 25 generations based on the recorded successful Cr values since the last $\mathrm{Cr_m}$ update. Using fuzzy logic controllers, Liu and Lampinen [11] presented a fuzzy adaptive DE, whose inputs incorporated the relative function values and individuals of successive generations to adapt the parameters for mutation and crossover. Brest et al. [12] proposed a self-adaptation scheme for the DE control parameters, known as jDE. The control parameters were adjusted by means of evolution of F and Cr. In jDE, a set of F and Cr values was assigned to each individual in the population, augmenting the dimensions of each vector. Zhang et al. [13] introduced an adaptive DE algorithm with optional external memory (JADE). In it, at each generation, the crossover probability Cr_z of each individual $\mathrm{x_z}$ was independently generated according to a normal distribution of mean μCr and standard deviation of 0.1. μCr was initialized at a value of 0.5 and updated. Similarly, F_z of each individual x_z was independently generated according to a Cauchy distribution with a location parameter μF and a scale parameter 0. The location parameter μF was initialized to 0.5 and subsequently updated at the end of each generation. Das *et al.*[14] introduced two versions for adapting F in DE. In the first scheme, F was randomly chosen between 0.5 and 1.0, while in the second scheme, F was initialized with a value of 1.0, and then linearly reduced to 0.1 during the evolution process. Generally speaking, such techniques may need adapting other parameters which may affect the performance of DE.

In this paper, we have evolved two DE parameters (such as F and Cr) by using three different algorithms. They are: (1) DE (this variant is recognized as Var1); (2) covariance adaptation matrix evolution strategy (CMA-ES)[15] (Var2); and (3) GA (Var3). That means, we are applying DE to solve the optimization problems, and within DE, we are using one of the above three algorithms to self-adaptively select DE parameters. The performances of these variants are compared to each other as well as to a DE with a single set of parameter values. From the results obtained, it is clear that the self-adaptive mechanism is better than a DE with a fixed set of parameters. Among the three variants, Var2 is the best considering the best fitness values found, while based on the average fitness values, Var3 is the best. However, these two variants are computational expensive in comparison with Var1.

The rest of this paper is organized as follows: section 2 presents and overview of DE. Section 3 discusses the self-adaptive mechanisms used in this paper, while section 4 presents the computational results. Finally, conclusions are elaborated in section 5.

2 Differential Evolution

DE uses the concept of a larger population from a GA and self-adapting mutation from ES [2], and differs from traditional EAs mainly in its generation of new vectors which adds the weighted difference vector (DV) between two individuals to a third individual [16]. It performs well when the feasible patches are parallel to the axes [17] but converges prematurely when dealing with a multi-modal fitness function because it loses its diversity [18, 19].

2.1 Mutation

The simplest form of this operation is that a mutant vector is generated by multiplying an amplification factor, F, by the difference between two random vectors, with the result added to a third random vector (DE/rand/1) [20] as:

$$\vec{x}_{z,t} = \vec{x}_{r_1,t} + F\left(\vec{x}_{r_2,t} - \vec{x}_{r_3,t}\right) \tag{1}$$

where r_1, r_2, r_3 are random numbers (1,2, ..., PS), $r_1 \neq r_2 \neq r_3 \neq z$, x a decision vector, F a positive control parameter for scaling the DV and *t* the current generation.

This operation enables DE to explore the search space and maintain diversity and there are many strategies for it, such as DE/best/1 [20], DE/current-to-best/1[21]. For more details, readers are referred to [22].

2.2 Crossovers

The DE family of algorithms usually depends on two crossover schemes, exponential and binomial, which are briefly discussed below.

In an exponential crossover, firstly, an integer, l, is randomly chosen within the range [1, D] and acts as the point in the target vector from where the crossover or exchange of components with the donor vector starts. Another integer, L, chosen from interval [1, D] denotes the number of components the donor vector actually contributes to the target. After the generation of l and L, the trial vector is obtained as:

$$u_{z,j,t} = \begin{cases} v_{z,j,t} & for\ j = \langle l \rangle_D, \langle l+1 \rangle_D, \ldots, \langle l+L-1 \rangle_D \\ x_{z,j,t} & for\ all\ other\ j \in [1, D] \end{cases} \tag{2}$$

where $j = 1,2, \ldots, D$, and the angular brackets, $\langle l \rangle_{D,}$ denote a modulo function with a modulus of D and starting index of l.

The binomial crossover is performed on each of the j^{th} variables whenever a randomly chosen number (between 0 and 1) is less than or equal to the crossover rate, Cr. In this case, the number of parameters inherited from the donor has a (nearly) binomial distribution as:

$$u_{zj,t} = \begin{cases} v_{zj,t}, & if\ (rand \leq Cr\ or\ j = jrand) \\ x_{zj,t}, & otherwise \end{cases} \tag{3}$$

where $rand \in [0,1]$ and $j_{rand} \in [1,2, \ldots, D]$ is a randomly chosen index which ensures that $\vec{u}_{z,t}$ receives at least one component from $\vec{v}_{z,t}$.

2.3 Selection

An offspring will be selected if it is better than its parent.

3 Self-adaptive DE Variants Based on EAs

To start with, in this paper, Table 1 shows the general framework of DE used in this paper.

Table 1. General framework of DE used in this paper

STEP 1: At generation $t = 1$, generate an initial random population of size PS. The variables of each individual (z) must be within a range such as:

$$x_{z,j} = \underline{x}_{z,j} + rand \times (\overline{x}_{z,j} - \underline{x}_{z,j})$$

where $\underline{x}_{z,j}, \overline{x}_{z,j}$ are the lower and upper bounds of the decision variable x_j, and *rand* is a random number, $rand \in [0,1]$.

STEP 2: Generate initial values for F and Cr

STEP 3: Evolve F and Cr using an EA, as shown in **3.1, 3.2** and **3.3**

STEP 4: Generate new offspring as follows:

4.1 Generate the offspring vector $\vec{u}_z$, using DE/current-to-best/bin and the corresponding F_z and Cr_z obtained in **Step 3** such as

$$u_{zj,t} = \begin{cases} x_{i,j} + F_z.\left(\left(x_{r_1,j} - x_{r_2,j}\right) + \left(x_{best,j} - x_{i,j}\right)\right), & if\ (rand \le Cr_z\ or\ j = jrand) \\ x_{z,j}, & otherwise \end{cases}$$

where r_1 and r_2 are random integer numbers $\in [1, PS]$ and both are not similar to i

4.2 Update $\boldsymbol{F_z}$ and $\boldsymbol{Cr_z}$, if required, see **3.1**

STEP 8: Stop if the termination criterion is met; **else,** set $t = t + 1$ **and** go to **STEP 3**.

In this paper, three variants are used to self-adaptively generate F and Cr, as described below.

3.1 Var1: Adapting F and Cr Using DE

Here, Cr and F are self-adaptively calculated using a simple DE algorithm, as follows:

- At $t = 1$, each individual in PS is assigned with $\dot{F}_z$ and $\dot{Cr}_z$, where $\dot{F}_z = N(0.5,0.1)$ and $\dot{Cr}_z = N(0.5,0.1)$. If the value is less than 0.01 or larger than 1.0, it is reflected back to be between to 0.01 and 1, respectively.
- Then, both parameters are calculated as follows:

$$F_z = \begin{cases} \dot{F}_{r_1} + rand_1 \times \left(\dot{F}_{r_2} - \dot{F}_{r_3}\right), & if\ (rand_2 < \tau_1) \\ rand_3 & otherwise \end{cases} \quad (4)$$

$$Cr_z = \begin{cases} \dot{Cr}_{r_1} + rand_4 \times \left(\dot{Cr}_{r_2} - \dot{Cr}_{r_3}\right), & if\ (rand_2 < \tau_1) \\ rand_5 & otherwise \end{cases} \quad (5)$$

where $rand_\Gamma \in [0,1]\ \forall\, \Gamma = 1,2 \ldots,5$ and $\tau_1 = 0.75$. If the value is less than 0.01 or larger than 1, it is truncated to 0.1 and 1, respectively.

- If the new offspring is better than its parent, then $\dot{F}_z = F_z$ and $\dot{Cr}_z = Cr_z$.

3.2 Var2: Adapting F and Cr Using CMA-ES

Here, Cr and F are self-adaptively calculated using CMA-ES, such that

- At $t = 1$, $\vec{x}^t_{mean}$ is initialized as $N(0.5,0.1,np)$, where np is 2, $\vec{x}^t_{1,1}$ refers to F_z, while $\vec{x}^t_{1,2}$ refers to Cr_z.
- A new population, which represents both F and Cr, is generated using CMA-ES:

$$\overrightarrow{xp}^{t+1}_{z=1:PS} = \vec{x}^t_{mean} + \sigma^t B^t Q^t G_{z=1:PS} \tag{6}$$

  where $G_{k=1:PS}$ are independent realizations of a D-dimensional standard normal distribution with zero-mean and a covariance matrix equal to the identity matrix I. These base points are rotated and scaled by the eigenvectors B^t and the square root of the eigenvalues Q^t of the covariance matrix C^t. The C^t, and the global step-size σ^t are continuously updated after each generation *t* [15].
- Then, both parameters are calculated as follows:

$$F_z = \begin{cases} xp_{z,1}, & if\ (rand_1 < \tau_1) \\ rand_2 & otherwise \end{cases} \tag{7}$$

$$Cr_z = \begin{cases} xp_{z,2}, & if\ (rand_1 < \tau_1) \\ rand_3 & otherwise \end{cases} \tag{8}$$

- Then $\vec{x}^t_{mean}$, and all other CMA-ES's parameters are updated as suggested in https://www.lri.fr/~hansen/cmaes.m. It is worthy to mention here that, to measure the quality of each $\overrightarrow{xp}_z$, the objective function of the corresponding $\vec{x}_z$ is used instead.

3.3 Var3: Adapting F and Cr Using GA

Here, a multi-parent crossover GA (MPC-GA) [23] is used to evolve DE parameters.

- At $t = 1$, an initial population (xp) of $\dot{F}_z$ and $\dot{Cr}_z$ is initialized, where $xp_{z,1:2} = N(0.5,0.1)$.
- Then an archive pool (A_{arch}) is filled with the best *m* individuals (based objective function of the corresponding $\vec{x}_z$.
- Then a tournament selection procedure, with size tc, takes place, from which the best individual is chosen and saved in the selection pool.
- for each three consecutive individuals in the selection pool, three offspring are generated as

$$\vec{y}_1 = \overrightarrow{xp}_1 + \beta \times (\overrightarrow{xp}_2 - \overrightarrow{xp}_3) \tag{9}$$

$$\vec{y}_2 = \overrightarrow{xp}_2 + \beta \times (\overrightarrow{xp}_3 - \overrightarrow{xp}_1) \quad (10)$$

$$\vec{y}_3 = \overrightarrow{xp}_3 + \beta \times (\overrightarrow{xp}_1 - \overrightarrow{xp}_2) \quad (11)$$

where $\beta = N(0.7,0.1)$ [23].

- On each generated $\vec{y}_z$, a diversity operator is applied. In it, for each individual a uniform random number $\in$ [0, 1] is generated, if it is less than a predefined probability, $p = 0.1$, then $y_z^j = x_{arch}^j$.
- Subsequently, set $xp = y$
- Then, both parameters are calculated as follows:

$$F_z = \begin{cases} xp_{z,1}, & if\ (rand_1 < \tau_1) \\ rand_2 & otherwise \end{cases} \quad (12)$$

$$Cr_z = \begin{cases} xp_{z,2}, & if\ (rand_1 < \tau_1) \\ rand_3 & otherwise \end{cases} \quad (13)$$

4 Experimental Results

In this section, a comparison among all variants is elaborated by solving a set of problems presented in the CEC2014 competition on real-parameter optimization [24], which contains 30 test problems with 30 dimensions, with the following mathematical properties: F01-F03 are unimodal functions, F04-F16 are simple multimodal functions, F17-F22 are hybrid functions, while F23-F30 are composition functions.

To add to this, all variants were compared with a DE, as shown in equation 4, with fixed values of its parameters, such that F=0.5, Cr=0.5, while PS was set to 100 individuals for all variants and τ_1=0.75. All variants were run 51 times for each test problem, where the stopping criterion was to run for up to 10,000D FEs. The algorithm was coded using Matlab R2012b, and was run on a PC with a 3.4 GHz Core I7 processor with 16 GB RAM, and windows 7.

To begin with, the best results obtained by all variants are shown in Table 2. From these results, it is clear that all variants were better than DE for F01, with the consideration that Var1 performs in F01, while all variants were able to obtain the same values for F02 and F03. For the multi-modal test problems, Var2 was the best for most of the test problems. However, Var1 was superior to all other variants for F05 and F13 and F15, while DE was the best in F14. In regards to the hybrid function, Var2 was the best for four test problems, while Var1 was the best for only F22 and Var3 performed best in F21. For the composition functions, Var2 performed best for 6 test functions. DE and Var1 obtained the same best result in F26, while Var3 was the best in F29.

Considering the average results obtained by all variants, see Table 3, it is noticed that Var3 was the best for the unimodal test functions, followed by Var1. However, Var2 was the worst variant for those test problems. For the multi-modal test functions,

Table 2. Best Results obtained by all variants

Prob.	DE1	Var1	Var2	Var3
F01	2.604E+05	**1.276E+03**	2.422E+03	6.483E+03
F02	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
F03	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
F04	1.441E+01	1.012E-04	**0.000E+00**	**0.000E+00**
F05	2.074E+01	**2.000E+01**	2.022E+01	2.042E+01
F06	2.259E-04	4.765E-01	**0.000E+00**	9.643E-02
F07	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
F08	1.333E+01	**0.000E+00**	**0.000E+00**	1.417E+01
F09	1.199E+02	3.283E+01	**2.098E+01**	3.310E+01
F10	1.955E+02	9.155E+00	**1.767E+00**	4.642E+02
F11	5.248E+03	1.942E+03	**1.348E+03**	2.148E+03
F12	1.506E+00	2.782E-01	**2.427E-01**	4.764E-01
F13	1.416E-01	**1.407E-01**	1.744E-01	1.578E-01
F14	**1.065E-01**	1.286E-01	1.503E-01	1.842E-01
F15	1.070E+01	**3.422E+00**	3.698E+00	5.246E+00
F16	1.084E+01	1.065E+01	**8.485E+00**	1.025E+01
F17	1.790E+04	8.806E+02	**2.211E+02**	6.994E+02
F18	1.893E+02	4.807E+01	**3.225E+01**	4.820E+01
F19	4.091E+00	5.492E+00	**2.604E+00**	4.365E+00
F20	4.888E+01	4.376E+01	**1.423E+01**	1.540E+01
F21	2.105E+03	3.068E+02	2.219E+02	**1.471E+02**
F22	5.073E+01	**2.544E+01**	2.592E+01	3.123E+01
F23	**3.152E+02**	**3.152E+02**	**3.152E+02**	**3.152E+02**
F24	2.232E+02	2.235E+02	**2.217E+02**	2.233E+02
F25	2.027E+02	2.033E+02	**2.026E+02**	2.027E+02
F26	**1.001E+02**	**1.001E+02**	1.002E+02	1.001E+02
F27	3.038E+02	3.745E+02	**3.000E+02**	3.004E+02
F28	7.202E+02	7.169E+02	**6.494E+02**	7.500E+02
F29	8.856E+02	5.318E+02	7.195E+02	**4.822E+02**
F30	6.329E+02	9.568E+02	**5.052E+02**	7.080E+02

Var1, Var2, Var3 and DE were able to obtain the best results for 6, 3, 3, 1 test function (s), respectively. For the hybrid functions, Var2 was superior to all other variants for 5 test functions, while Var2 performed best for F22. Considering the composition functions, DE, Var1 and Var3 were able to obtain the same result in F23. Var1 and Var3 obtained the same result in F26, while DE was the best for 3 test functions, while Var3 was the best for F28, F29 and F30.

To continue our analysis, the average computational time for each variant were calculated. The computational time was calculated as the average time consumed to reach the best known solutions with an error 1.0E-08, i.e. the stopping criteria is $[f(\vec{x}) - f(\vec{x^*}) \leq 1.0E-08]$, where $f(\vec{x^*})$ is the best known solution. The summary results are shown in Table 4. From this table, Var1 was the best.

Similarly, the average numbers of fitness evaluations to reach the above mentioned stopping criterion were recorded, see Table 5. From that table, it is found that Var3 performed best.

Table 3. Average Results obtained by all variants

Prob.	DE	Var1	Var2	Var3
F01	1.160E+06	6.398E+04	3.979E+06	**3.865E+04**
F02	2.004E+03	**0.000E+00**	1.746E+07	**0.000E+00**
F03	**0.000E+00**	**0.000E+00**	3.886E+02	**0.000E+00**
F04	1.147E+02	1.719E+01	8.887E+01	**1.578E+00**
F05	2.092E+01	**2.000E+01**	2.046E+01	2.056E+01
F06	**1.132E+00**	1.593E+01	4.088E+00	2.292E+00
F07	1.335E-02	1.090E-02	5.373E-01	**5.987E-03**
F08	6.529E+01	**1.615E-04**	1.023E+01	2.000E+01
F09	1.444E+02	5.552E+01	**5.425E+01**	6.272E+01
F10	2.705E+03	**1.272E+02**	4.348E+02	7.950E+02
F11	6.140E+03	**2.782E+03**	2.974E+03	3.499E+03
F12	2.043E+00	**4.404E-01**	7.345E-01	7.981E-01
F13	2.582E-01	2.625E-01	2.561E-01	**2.535E-01**
F14	2.511E-01	**2.310E-01**	2.465E-01	2.546E-01
F15	1.301E+01	7.306E+00	**6.854E+00**	6.903E+00
F16	1.183E+01	1.130E+01	**1.036E+01**	1.103E+01
F17	8.314E+04	1.810E+03	9.054E+03	**1.403E+03**
F18	3.653E+02	1.323E+02	1.488E+02	**1.265E+02**
F19	6.492E+00	1.149E+01	7.627E+00	**6.340E+00**
F20	8.107E+01	1.743E+02	6.582E+01	**2.909E+01**
F21	3.953E+03	8.281E+02	3.149E+03	**3.964E+02**
F22	2.504E+02	1.941E+02	**1.416E+02**	1.499E+02
F23	**3.152E+02**	**3.152E+02**	3.157E+02	**3.152E+02**
F24	**2.285E+02**	2.339E+02	2.289E+02	2.290E+02
F25	**2.038E+02**	2.088E+02	2.062E+02	2.048E+02
F26	1.081E+02	**1.042E+02**	1.081E+02	**1.042E+02**
F27	**3.935E+02**	4.464E+02	4.192E+02	4.021E+02
F28	8.876E+02	9.502E+02	9.049E+02	**8.631E+02**
F29	2.155E+05	3.839E+05	1.666E+05	**8.067E+02**
F30	1.978E+03	2.388E+03	3.896E+03	**1.786E+03**

Table 4. Computational time, in seconds, of each variant

DE1	Var1	Var2	Var3
18.35	**14.58**	24.81	20.59

Table 5. Average number of fitness evaluations

DE2	Var1	Var2	Var3
289259	281778.3	287584	**280234**

To study the statistical difference between any two stochastic algorithms, a non-parametric test, Wilcoxon Signed Rank Test [25], is chosen. As a null hypothesis, it is assumed that there is no significant difference between the best and/or mean values of two samples. Whereas the alternative hypothesis is that there is a significant difference in the best and/or mean fitness values of the two samples, with a significance level of 5%. Based on the test results, one of three signs (+, −, and ≈) is assigned for the comparison of any two algorithms (shown in the last column), where the "+" sign means the first algorithm is significantly better than the second, the "−" sign means that the first algorithm is significantly worse, and the "≈" sign means that there is no

significant difference between the two algorithms. The results are shown in **Table 6**. From this table, considering the best results, it is found that all variants are statistically better than DE. To add to this, Var2 is statistically better than all variants. In regards to the average results, it is interesting to find that Var3 was statically better than all variants, while there were no significant differences among other variants.

Table 6. The Wilcoxon non-parametric test among all variants, based on the best and averge fitness values obtained, where × means that no statisitcal test was applicable

Variants	Best fitness value				Average fitness value			
	DE	Var1	Var2	Var3	DE	Var1	Var2	Var3
DE	×	−	−	−	×	≈	≈	−
Var1 (DE & DE)	+	×	−	≈	≈	×	≈	−
Var2 (DE & CMA-ES)	+	+	×	+	≈	≈	×	−
Var3 (DE & GA)	+	≈	−	×	+	+	+	×

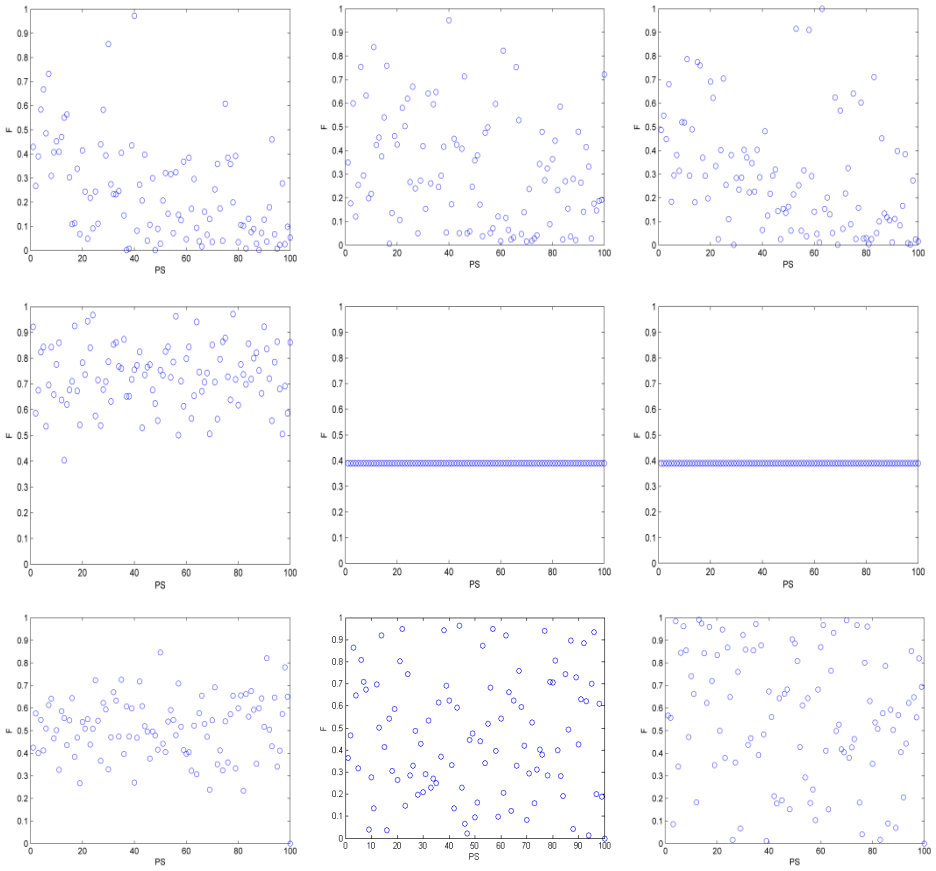

Fig. 1. F values after 2, 100 and 500 generations. The 1st, 2nd and 3rd are of Var1, Var2 and Var3, respectively.

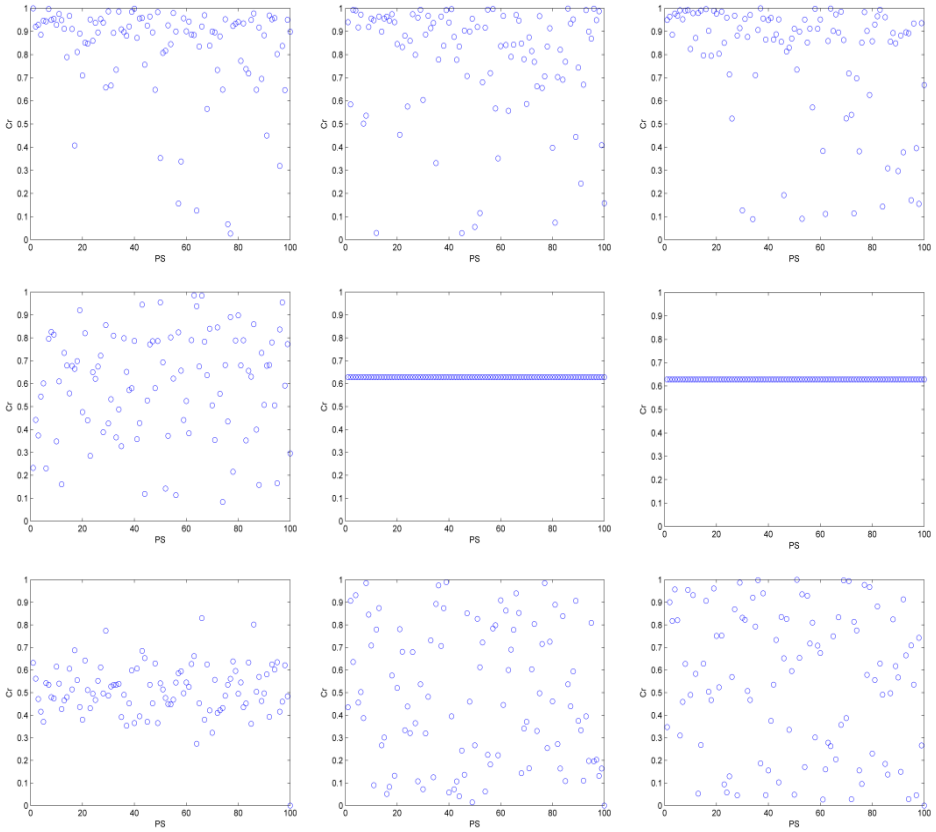

Fig. 2. Cr values after 2, 100 and 500 generations. The 1st, 2nd and 3rd are of Var1, Var2 and Var3, respectively.

It is also important to investigate why DE average results are not consistent with the best results when using CMA-ES in adapting DE parameters (in Var2)? To do this, we plotted the *F* and *Cr* values after 2, 100 and 500 generations. We found that CMA ES used to convergence to a single value for each parameter, see Fig. 1 and Fig. 2, and this combination of parameters might not be the best during the entire evolutionary process. In contrast, both Var1 and Var3 maintained good diversity.

5 Conclusions and Future Work

There is no doubt that the success of DE depends on its parameters. However, the selection of its parameters is not a simple task. This motivated many researchers to investigate this direction. Although many techniques were proposed to self-adaptively adapt its parameters, using EAs to do this process have not been fully explored. Consequently, in this paper, we compared the performance of DE when adapting its parameters using three different EAs (DE, CMA-ES and GA). From the results,

we found that adapting DE's parameters using CMA-ES had the ability to obtain the best results, in terms of the best solutions obtained, in many occasion. However, its average results were inferior to the variant that considered GA.

Although, DE, with a single combination of parameters, was good in several occasions, there is a question how can we determine it? To add to this, it was interesting to find that using EAs to adapt DE's parameters might save computational time; this is because it had the ability to quickly find the optimal solutions in many occasions.

There are many open directions can be done in this direction. For instance, recent years have shown much interest in developing multi-operator DE algorithms. Doing the same trend and use multi-operator algorithms to evolve DE's parameters may be a possible future work. To add to this, providing a detailed comparison to other adaptation techniques is also important.

References

1. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, MA (1989)
2. Storn, R., Price, K.: Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces, in Technical Report, International Computer Science Institute (1995)
3. Rechenberg, I.: Evolutions strategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution. Fromman-Holzboog, Stuttgart (1973)
4. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
5. Abbass, H.A.: The self-adaptive Pareto differential evolution algorithm. In: IEEE Congress on Evolutionary Computation (2002)
6. Elsayed, S.M., Sarker, R.A., Essam, D.L.: Multi-operator based evolutionary algorithms for solving constrained optimization Problems. Computers and Operations Research 38(12), 1877–1896 (2011)
7. Elsayed, S.M., Sarker, R.A., Essam, D.L.: Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems. In: IEEE Congress on Evolutionary Computation (2011)
8. Elsayed, S.M., Sarker, R.A., Essam, D.L.: Improved genetic algorithm for constrained optimization. In: 2011 International Conference on Computer Engineering & Systems, ICCES (2011)
9. Elsayed, S.M., Sarker, R.A., Essam, D.L.: On an evolutionary approach for constrained optimization problem solving. Applied Soft Computing 12(10), 3208–3227 (2012)
10. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization. IEEE Transactions on Evolutionary Computation 13(2), 398–417 (2009)
11. Liu, J., Lampinen, J.: A Fuzzy Adaptive Differential Evolution Algorithm. Soft Computing - A Fusion of Foundations. Methodologies and Applications 9(6), 448–462 (2005)
12. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (2006)
13. Zhang, J., Sanderson, A.C.: JADE: Adaptive Differential Evolution With Optional External Archive. IEEE Transactions on Evolutionary Computation 13(5), 945–958 (2009)

14. Das, S., Konar, A., Chakraborty, U.K.: Two improved differential evolution schemes for faster global search. In: The 2005 Conference on Genetic and Evolutionary Computation, pp. 991–998. ACM, Washington, DC (2005)
15. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (2001)
16. Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (1997)
17. Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC, competition and special session on single objective constrained real-parameter optimization. 2010: Technical Report, Nangyang Technological University, Singapore (2010)
18. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: 6th Int. Mendel Conference on Soft Computing, Brno, Czech Republic (2000)
19. Vesterstrom, J., Thomsen, R.: A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: IEEE Congress on Evolutionary Computation (2004)
20. Storn, R.: On the usage of differential evolution for function optimization. In: Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS (1996)
21. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential evolution: A practical approach to global optimization. Natural Computing Series. Springer, Berlin (2005)
22. Das, S., Suganthan, P.N.: Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation 15(1), 4–31 (2011)
23. Elsayed, S.M., Sarker, R.A., Essam, D.L.: GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems. In: IEEE Congress on Evolutionary Computation (2011)
24. Liang, J.J., Qu, B.-Y., Suganthan, P.N.: Problem Definitions and Evaluation Criteria for the CEC, Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Computational Intelligence Laboratory and Nanyang. Technological University, China and Singapore (2014)
25. Corder, G.W., Foreman, D.I.: Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach. John Wiley, Hoboken (2009)