A Study on Self-configuration in the Differential Evolution Algorithm

Rodrigo C. P. Silva*, Rodolfo A. Lopes†, Alan R. R. Freitas‡ and Frederico G. Guimarães§
*Department of Electrical and Computer Engineering, McGill University, Montreal, Canada
Email: rodrigo.silva@mail.mcgill.ca
†Graduate Program in Electrical Engineering, Federal University of Minas Gerais (UFMG), Belo Horizonte - MG, Brazil
Email: rodolfo.ufop@gmail.com
‡Department of Computer Science, Federal University of Ouro Preto (UFOP), Ouro Preto - MG, Brazil
Email: alandefreitas@iceb.ufop.br
§Frederico G. Guimarães
Department of Electrical Engineering, Federal University of Minas Gerais (UFMG), Belo Horizonte - MG, Brazil
Email: fredericoguimaraes@ufmg.br

***Abstract*—The great development in the area of evolutionary algorithms in recent decades has increased the range of applications of these tools and improved its performance in different fronts. In particular, the Differential Evolution (DE) algorithm has proven to be a simple and efficient optimizer in several contexts. Despite of its success, its performance is closely related to the choice of variation operators and the parameters which control these operators. To increase the robustness of the method and the ease of use for the average user, the pursuit for methods of self-configuration has been increasing as well. There are several methods in the literature for setting parameters and operators. In order to understand the effects of these approaches on the performance of DE, this paper presents a thorough experimental analysis of the main existing paradigms. The results show that simple approaches are able to bring significant improvements to the performance of DE.**

I. Introduction

Differential Evolution (DE)[1] can be classified as a member of the broader class of population-based evolutionary algorithms (EAs) and it is an important method for global optimization. The reasons for its success can be found in its simplicity and ease of implementation, while at the same time demonstrating reliability and high performance [2]. It has proven to be successful in a wide range of problems over continuous [3], discrete [4], [5] and mixed spaces [6].

The DE has 3 control parameters: the population size NP, the scale factor of the perturbations generated by mutation F and the crossover rate CR, in addition, it has also a number of different strategies for offspring generation [7], [8]. In one hand, these features help making the DE robust with respect to the diverse situations cited above, giving to it the tools to tackle the specificities of each problem. On the other hand, they are also responsible for adding an extra difficulty to its use. The definition of which of the available strategies to be applied to the problem at hand is not trivial.

To make DE easier to use and more robust to the different problems, self-adaptation and self-configuration have become very important topics in the DE community. The basic idea is to leave to the algorithm itself the responsibility of selecting a suitable set of parameters to the problem in hands. This basic idea has culminated in the publication of a large variety of self-configured versions of the DE. All of them work on at least one of these two fronts: (i) automatic parameter setting [9], [10], [11], [12] and (ii) automatic selection of variation operators [13], [14], [15]. Although all these approaches present improvements with respect to the basic differential evolution, the effect of the proposed mechanisms, applied in each front and their interaction does not seem to be clear. The main problem is that in addition to the self-configuration mechanisms, other changes are made to the DE itself. For this reason it is impossible to tell what part of the algorithm is really making the difference for its overall performance.

In order to understand the solely effect of the self-configuration approaches, this paper presents a thorough experimental analysis of the main existing paradigms. Different from other papers in the field, which compare the different algorithms, in this paper the various approaches are detached from the original algorithm and implemented within the same basic framework. In this way, the effect of each configuration strategy can be isolated and a fair comparison, disregarding all other algorithmic specificities, can be made. This study is performed on different mechanisms for parameter setting (front (i)) such as, self-adaptation, adaptation and randomization and different mechanisms for selection of mutation operators (front (ii)) namely, probability matching and Q-learning. In addition of showing what is the impact of each mechanism in the DE performance we also show how these mechanisms interact. The results indicate that even the simplest strategies are able to improve the basic DE and match the performance of most complex ones.

II. The Differential Evolution Algorithm (DE)

As the majority of evolutionary algorithms, the original DE algorithm[1] [1] starts with a population of candidate solutions randomly generated within the domain region of the problem, usually described as:

[1]See [7], [3] for additional details.

$$\mathcal{X} = \left\{ \mathbf{x} \in \mathbb{R}^D : x_k^{min} \leq x_k \leq x_k^{max}, k = 1, ..., D \right\} \quad (1)$$

where $x_k^{\min}$ and $x_k^{\max}$ are respectively the lower and upper limits of each variable and D is the problem dimension, i.e., the number of variables in the problem.

In this paper, $x_{g,i,j}$ represents the individual with index $i \in [1, NP]$ in the population, during the generation g. $j = 1, \ldots, D$ represents the design parameter index. A given individual is then represented by:

$$\mathbf{x}_{g,i} = \left\langle x_{g,i,1}, x_{g,i,2}, x_{g,i,3}, ..., x_{g,i,D} \right\rangle \quad (2)$$

After initialization, for each individual in the population a new individual $\mathbf{v}_{g,i}$, called mutant, is generated by a mutation operator. In Differential Evolution mutation is based on the difference between individuals randomly chosen from the current population as follows:

$$\mathbf{v}_{g,i} = \mathbf{x}_{g,r1} + F(\mathbf{x}_{g,r2} - \mathbf{x}_{g,r3}) \quad (3)$$

where $r_1 \neq r_2 \neq r_3 \in \{1, \ldots, NP\}$ are mutually distinct random indexes and F the scale factor applied to the differential vector.

After mutation a trial vector $\mathbf{u}_{g,i}$ is produced through recombination of $\mathbf{x}_{g,i}$ and $\mathbf{v}_{g,i}$. In the basic DE algorithm, the discrete recombination with probability CR is used, as we can see in the following scheme:

$$\mathbf{u}_{g,i,j} = \begin{cases} \mathbf{v}_{g,i,j}, & \text{if } \mathcal{U}_{[0,1]} \leq CR \\ \mathbf{x}_{g,i,j}, & \text{otherwise} \end{cases} \quad (4)$$

where $\mathcal{U}_{[a,b]}$ represents the sampling of a random variable with uniform distribution in the interval $[a, b]$. In this way, F and CR represent control parameters of the algorithm.

Finally, the trial vector $\mathbf{u}_{g,i}$ competes with the current solution $\mathbf{x}_{g,i}$ based on their objective function evaluations. The trial vector replaces the current solution if it is better than the current solution or if it has the same value of objective function. This process can be described by the equation below:

$$\mathbf{x}_{g+1,i} = \begin{cases} \mathbf{u}_{g,i} & \text{if } f(\mathbf{u}_{g,i}) \leq f(\mathbf{x}_{g,i}) \\ \mathbf{x}_{g,i} & \text{otherwise} \end{cases} \quad (5)$$

In DE, most of the responsibility of adapting the mutation step size is left to the distribution of the solutions itself [3]. If the solutions are close to each other the magnitude of the difference vector in Equation 3 is going to be small. The opposite is going to happen if the solutions are far from each other.

Even though the scale factor F just controls the relative step size, provide multiple Fs during the search course has some potential benefits. One of them is that with different values of F the number of possible difference vectors increases, what in turn, increases the exploration power of DE and avoids stagnation [16].

The crossover control parameter CR is linked to the number of components inherited from the mutant vector. As shown in [3], CRs of small magnitude ($0 \leq CR \leq 0.2$) are more interesting for the optimization of separable functions. In this scenario, each variable can be optimized independently. On the other hand, for non-separable problems high values of CR are more suitable. In this situation there is a correlation among the variables and because of that it is more appropriate to optimize them simultaneously.

Besides of these control parameters DE can also be modified by the use of different variation operators (mutation + crossover) . In [8], an empirical comparison among various of these operators (See some definitions at Section IV) for the creation of trial vector is made. The authors show that there are some strategies which favor exploration, others exploitation, some of them work better in multimodal problems and others in uni-modal problems.

III. Self-Configuration in DE

As we could see, the basic DE presented above is just a framework and it can be amended in various ways to tackle the specificities of each different problem. If in one hand this great number of possibilities can be seen as an advantage, on the other hand, its use by the inexperienced user can be difficult. Even for experienced users, the adequate configuration of the algorithm usually demands a lot of time doing preliminary experiments and/or; information about the problem (e.g multimodality, separability) which is not always available.

In this context, what is left to the user is the use of a standard configuration what can lead to a poor performance of the algorithm. Therefore, to turn DE into an easy-to-use general optimization tool some self-configuration ability must be provided.

In the literature, one can find plenty of approaches and mechanisms which avoid interaction with the user for DE configuration. In these works the problem of self-configuration is tackled in two fronts. The first handles the control parameter configuration and the second one the configuration of variation operators (mutation + crossover). In the following sections some selected approaches for each front will be presented. The goal was to choose approaches with mechanisms as diverse as possible in order to access the effect of each paradigm in the performance of the algorithm. Although this selection is not exhaustive, these paradigms are all present in state-of-the-art DE algorithms, such as, SaDE [11], [17], JADE [18], jDE [9], CoDE [19] and PM-AdapSS-DE [13].

A. Parameter Setting

1) Randomization: One of the simplest ways to take the responsibility of parameter setting off the user is the pure randomization of the control parameters. Equation 6 presents a procedure commonly used in the literature (see, [20],[9],[21],[22]and [23]).

$$P'_{g,i} = \mathcal{U}_{[a,b]} \text{ , if } \mathcal{U}_{[0,1]} \leq \tau \quad (6)$$

P' is the control parameter generated by this approach which typically represents F and CR. $\mathcal{U}_{[a,b]}$ represents the sampling of a uniformly distributed random variable in the interval $[a, b]$

and $\tau \in [0, 1]$ represents the odds for that parameter to be re-sampled.

Another common approach is to do the sampling using a Gaussian distribution as follows:

$$P'_{g,i} = \mathcal{N}_{[a,b]} \tag{7}$$

where, $\mathcal{N}_{[a,b]}$ is the sampling of a normally distributed random function with mean a and standard deviation b. This approach can be seen in [11], [3] and [24].

2) Randomization with Adaptation: This approach resembles the previous one, however, information from the underlying search process is used in order to generate new parameters. Equation 8 depicts the approach:

$$P'_{g,i} = \mathcal{N}_{[P_m,b]} \tag{8}$$

The main difference with respect to the previous one is the presence of a "learning" mechanism. It presents itself in the adaptation of the distribution mean (P_m). In [11] and [17], in a period of L_g generations a list L_P is made with the control parameter values (in this case just values of CR) which generated individuals which were better than their parents. Then, at every Lg generations the median of this list is used as the new mean for the Gaussian distribution (see Equation 9).

$$P_m = \text{median}(L_P) \tag{9}$$

In [18] and [25] again, a list of control parameters which have generated improved individuals is done. The difference is that now, there is a list for F, L_F and a list for CR, L_{CR}. F_m is computed by Equation 10 and CR_m by Equation 11.

$$F_m(g) = (1-c)F_m(g-1) + c(\text{lehmer}(L_F)) \tag{10}$$

$$CR_m(g) = (1-c)CR_m(g-1) + c(\text{mean}(L_{CR})) \tag{11}$$

where lehmer$(\cdot)$ is the Lehmer mean[2], mean$(\cdot)$ is the arithmetic mean and c is a control parameter of the algorithm.

3) Fixed Set of Strategies: This is a simple approach proposed in [19]. It consists in a set of $[F, CR]$ pairs which are chosen randomly for the generation of a new individual. The pairs have been selected in a way to provide the algorithm with different and complementary search strategies. The complete set of pairs is shown below:

1) $[F = 1.0, CR = 0.1]$
2) $[F = 1.0, CR = 0.9]$
3) $[F = 0.8, CR = 0.2]$

As mentioned at Section II, $[F = 1.0, CR = 0.1]$ is an interesting strategy for separable problems due to the small CR. $[F = 1.0, CR = 0.9]$ is useful for diversity maintenance and non-separable problems. Finally, $[F = 0.8, CR = 0.2]$ is considered to be good in exploitation.

[2]Consider $X = \{x_1, x_2, ..., x_n\}$, lehmer$(X) = \sum_{i=1}^{n} x_i^2 / \sum_{i=1}^{n} x_i$

4) Self-adaptation: In self-adaptation, the control parameters are encoded into the individual representation. Thereby, they also undergo the action of mutation and crossover operators. The rationale behind this approach is that control parameters which produce better individuals have a higher chance to survive. In this approach there is also a form of learning in which the control parameter values are gradually refined in accordance with the quality of the solutions they help to generate. This approach can be seen in [15] and [12]. The procedure for the generation of new parameter values is the DE itself as one can see in Equation 12.

$$\mathbf{P'}_{g,i,j} = \begin{cases} \mathbf{P}_{g,r1,j} + F'(\mathbf{P}_{g,r2,j} - \mathbf{P}_{g,r3,j}), & \text{if } \mathcal{U}_{[0,1]} \leq CR' \vee j = \delta_i \\ \mathbf{P}_{g,i,j}, & \text{otherwise} \end{cases} \tag{12}$$

where, $\mathbf{P}$ is a vector of control parameters. Although this approach presents its own control parameters F' and CR' which must be set, the sensibility of the algorithm's performance to them is usually low.

B. Configuration of Operators

The self-configuration of variation operators (mutation and crossover) has been studied in a number of publications in both Genetic Algorithms [26], [27], [28] and DE [17], [29], [30]. The idea behind this works is that the best variation operator depends on search process stage [14].

In all these works the configuration of operators can be divided in two parts:

1) Credit assignment: Defines a numerical measure of the operator's quality and the assignment policy;
2) Operator selection: Defines the operator selection policy based on the assigned credits.

There are various different approaches for both, credit assignment and operator selection. The comparison between different methods for credit assignment is out of scope of this paper. Below, the methods that will be used in the remainder of this paper are presented. They were chosen by their good performance reported in the literature.

As quality measure we will be using the improvement with respect to the parents [27];

$$c = of - pf \tag{13}$$

pf is the parent fitness and of is the offspring fitness.

As the assignment policy the Normalized Extreme Reward [31] will be used:

$$r_k = \frac{r'_k}{\max\limits_{b=1,...,K} r'_b}; \text{ and } r'_k = \max_{i=1,...,|C_k|} C_k(i) \tag{14}$$

where, r_k is the reward to be assigned to the operator $k \in 1, 2, ..., K$ and C_k is the set of credits c received by each application of the operator k.

Besides that, to update the reward vector r the procedure proposed by [32] has been used. By it, at each generation the computed r_k is stored in a queue of length $l = 10$. The actual reward the operator k is going to receive is the highest value in the queue. The rationale behind this is that it seems that the selection of operators which produce outlier individuals is better than selecting operators which produce good individuals in average [33].

Following we present the operator selection methods that will be compared in the computational experiments.

1) Probability Matching: Probability Matching, is one of the most used methods for operator selection. The basic idea behind it is, as its name suggests, to assign to each operator a probability of execution p_k proportional to its estimate of quality q_k. To this end, considering a set of k operator, this method keeps a vector of probabilities $P_t = \{p_{t,1}, ..., p_{t,k}\} (0 \leq p_{t,i} \leq 1; \sum_{i=1}^{k} p_{t,i} = 1)$ which is updated in the following way:

$$p_{t,k} = \frac{q_{t,k}}{\sum_{i=1}^{K} q_{t,i}} \tag{15}$$

After the probabilities computation, the operator k to be applied is defined by a Roulette Wheel algorithm. The application of this kind of procedure for the allocation of operators in DE can be seen in [14], [11] and [17].

2) Q-learning: In this approach the problem of allocation of operators is seen as a reinforcement learning problem. In the standard model of reinforcement learning [34] an agent interacts with the environment and receives information about its current state. Based on that information, the agent chooses an action to be executed. This action changes the environment which in turn returns a feedback sign to the agent. Through this sign, which can be a reward or a penalty, the agent "learns" how to behave.

For the operator allocation problem the DE is seen as an agent which has to learn what operator to apply at each step. It interacts with the population of individuals by means of the operators (mutation + crossover) and the feedback sign is generated by evaluating the population. If the used operators generate good individuals they must be reinforced.

One of the most used algorithms in reinforcement learning is called Q-learning [35]. In this paper we use a single state version of Q-learning presented in [36], with the Metropolis Criteria presented in [37]. Basically, the agent starts playing an action a and evaluates the consequences of this action by means of the received reward r as follows:

$$Q(a_t) \leftarrow \lambda(r + \gamma \max_{a_{t+1}} Q(a_{t+1})) + (1 - \lambda)Q(a_t) \tag{16}$$

$Q(a)$ is the reward expected by the execution of action a. $\max_{a_{t+1}} Q(a_{t+1})$ is an estimate of the maximum future reward and γ is a discount factor applied to it. Trying all actions repeatedly the algorithm learns the best execution policy based on the long term rewards and not only on the immediate ones. The method's pseudo code is depicted in Algorithm 1.

In practice what happens is that, in the beginning, the agent explores the action space almost randomly. The exploration rate is then gradually reduced (see Algorithm 1 line 7 and 16) and the agent starts to choose their actions based on what it "thinks" are the better options for that moment. The idea of treating the operator selection as a reinforcement learning problem is present in [38], [13] and [32].

Algorithm 1: Q-learning with Metropolis Criterion

```
1  Inputs:
2  $\lambda \leftarrow$ learning rate , $\gamma \leftarrow$ discount factor, $temperature \leftarrow 1$ ;
3  BEGIN
4  $\forall_a$ Q($a$) $\leftarrow$ 0;
5  $ar_t \leftarrow$ random action;
6  $ab_t \leftarrow \arg\max_\alpha Q(a_t)$;
7  if rand() < exp($(Q(ar_t) - Q(ab_t))/temperature$) then
8      $a_t \leftarrow ar_t$ ;
9  else
10     $a_t \leftarrow ab_t$;
11 end
12 Execute action $a_t$;
13 Compute reward $r$;
14 Q($a_t$) $\leftarrow \lambda(r + \gamma \max_{a_{t+1}}$Q($a_{t+1}$)) + (1 - $\lambda$)Q($a_t$);
15 $\lambda \leftarrow 0.99*\lambda$;
16 $temperature \leftarrow 0.98*temperature$;
17 goto 6;
18 END
```

IV. Self-configured Differential Evolution: Basic Structure

In order to test all these approaches and its combinations in Algorithm 2 we present a basic structure for a general Self-configured DE which can hold all the approaches. It works a "population" of control parameters to be updated by the procedures presented in Section III-A and a set of variation operators given by Equations 18, 19, 20 and 21. Each individual in the population of control parameters is defined as in Equation 17.

$$\mathbf{P}_{t,i} = \{F_1, CR_1, ..., F_k, CR_k\} \tag{17}$$

where $k \in 1, 2, 3, 4$ is the index of the variation strategy to which those parameters belong.

1) **rand/1/bin**

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{x}_{t,r1,j} + F(\mathbf{x}_{t,r2,j} - \mathbf{x}_{t,r3,j}), & \\ \quad \text{if } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i & \\ \mathbf{x}_{t,i,j}, & \text{otherwise} \end{cases} \tag{18}$$

2) **current-to-best/2**

$$\mathbf{u}_{t,i} = \mathbf{x}_{t,i} + Z(\mathbf{x}_{t,r1} - \mathbf{x}_{t,i}) + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) + F(\mathbf{x}_{t,r4} - \mathbf{x}_{t,r5}) \tag{19}$$

3) **rand/2/bin**

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{x}_{t,r1,j} + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) + F(\mathbf{x}_{t,r4} - \mathbf{x}_{t,r5}), & \\ \quad \text{if } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i & \\ \mathbf{x}_{t,i,j}, & \text{otherwise} \end{cases} \tag{20}$$

4) **current-to-rand/2**

$$\mathbf{u}_{t,i} = \mathbf{x}_{t,i} + Z(\mathbf{x}_{t,r1} - \mathbf{x}_{t,i}) + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) + F(\mathbf{x}_{t,r4} - \mathbf{x}_{t,r5}) \tag{21}$$

where $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \in \{1, \ldots, NP\}$ are mutually distinct random indexes, F is the scale factor applied to the differential vector and *best* is the index of the best individual in the current population.

Algorithm 2: Self-configured Differential Evolution

```
$t \leftarrow 1$
Initialize the Population $X_t\{\mathbf{x}_{t,i}; i = 1, 2, ..., NP\}$
Initialize the Population of Parameters $P_t = \{\mathbf{p}_{t,i} = \mathcal{U}_{[0,1]}; i = 1, 2, ..., NP\}$
Initialize the Quality Table $Q_t = \{\mathbf{q}_{t,i} = 0; i = 1, 2, ..., k\}$
Initialize the Reward Queue $W_{t,k} = \{\mathbf{r}_{t,i,j} = 0; i = 1, 2, ..., k, j = 1, 2, ..., S\}$
Evaluate $X_t$
while not stopping_condition do
  for $i = 1$ to $NP$ do
    Select the variation operators $o$
    Choose a random index $j \in NP$
    $\mathbf{p}'_{t,j,e} =$ Update($\mathbf{p}_{t,j,e}$)
    Generate a trial vector $\mathbf{u}_{t,i}$
        with the operators $o$
        and control parameters $\mathbf{p}'_{t,j,e}$
    Evaluate $\mathbf{u}_{t,i}$
    Compute the credits $c$
        due to the generation of the solution $\mathbf{u}_{t,i}$
    Selection
    if $\mathbf{u}_{t,i}$ is selected then
      $\mathbf{p}_{t,j,e} = \mathbf{p}'_{t,j,e}$
      Compute credit $c$
       due to the generation of the solution $\mathbf{u}_{t,i}$
    end if
  end for
  $\forall k$ Compute Rewards $r_k$ AND Update $W_{t,k}$
  Update$Q_t$
  $t \leftarrow t + 1$
end while
```

V. Experimental Setting

Section III-A presented different procedures for the self-configuration of parameters. As one can notice, with the exception of the approach based in a fixed set of strategies, all the the other ones present its own set of control parameters which were chosen from the related literature. In this way, with respect to the parameter setting the following approaches were implemented:

- Randomization with uniform distribution (**r**):

$$P'_{g,i} = \mathcal{U}_{[a,b]} \text{ , if } \mathcal{U}_{[0,1]} \leq \tau \tag{22}$$

  - $a = \begin{cases} 0 & \text{if} \quad P' = CR \\ 0.1 & \text{if} \quad P' = F \end{cases}$
  - $b = 1$
  - $\tau = 0.1$

- Randomization with a guided Gaussian (**gg**): As in [11] and [17] this procedure is applied just for the control parameter CR. For F just a simple randomization with Gaussian distribution is made.

$$\begin{aligned} CR'_{g,i} &= \mathcal{N}_{[P_m,a]} \\ F'_{g,i} &= \mathcal{N}_{[b,c]} \end{aligned} \tag{23}$$

  - $P_m = \text{median}(L_P)$
  - $L_g = 20$ generations; $a = 0.1$; $b = 0.5$; $c = 0.3$

- Fixed set of strategies (**co**):

$$[F', CR'] = \begin{cases} [1.0, 0.1] & \text{if} \quad \mathcal{U}_{[0,0.9]} \leq 0.3 \\ [1.0, 0.9] & \text{if} \quad 0.3 < \mathcal{U}_{[0,0.9]} \leq 0.6 \\ [0.8, 0.2] & \text{if} \quad 0.6 < \mathcal{U}_{[0,0.9]} \leq 0.9 \end{cases} \tag{24}$$

- Self-adaptation (**de**):

$$\mathbf{P}'_{t,i,j} = \begin{cases} \mathbf{P}_{t,r1,j} + F'(\mathbf{P}_{t,r2,j} - \mathbf{P}_{t,r3,j}), & \text{if } \mathcal{U}_{[0,1]} \leq CR' \vee j = \delta_i \\ \mathbf{P}_{t,i,j}, & \text{otherwise} \end{cases} \tag{25}$$

  - $F' = \mathcal{U}_{[0.6,1]}$
  - $CR' = \mathcal{U}_{[0.9,1]}$

Besides, as control experiments, 3 fixed pairs, and a randomized configuration (with values suggested in [3]) were used as well. They are:

- **px**: $[F = 1.0, CR = 0.1]$
- **py**: $[F = 1.0, CR = 0.9]$
- **pz**: $[F = 0.8, CR = 0.2]$
- **cs**: $[F = \mathcal{U}_{[0.4,1]}, CR = \mathcal{U}_{[0.9,1]}]$

With respect to the variation operator configuration we have:

- Q-learning (**q**)
  - $\gamma = 0.3$
- *Probability Matching* (**p**)

In both cases, the operator k just receives credits if it generates a solution that is better than its parent, otherwise, the assigned credit is 0. As control, we also use each variation strategy separately and a completely randomized selection as follows (the numbers refer to Equations 18, 19, 20 and 21, respectively):

Random (**r**):

$$\begin{cases} (1) & \text{if} \quad \mathcal{U}_{[0,1]} \leq 0.25 \\ (2) & \text{if} \quad 0.25 < \mathcal{U}_{[0,1]} \leq 0.5 \\ (3) & \text{if} \quad 0.5 < \mathcal{U}_{[0,1]} \leq 0.75 \\ (4) & \text{if} \quad 0.75 < \mathcal{U}_{[0,1]} \end{cases} \tag{26}$$

VI. Methodology

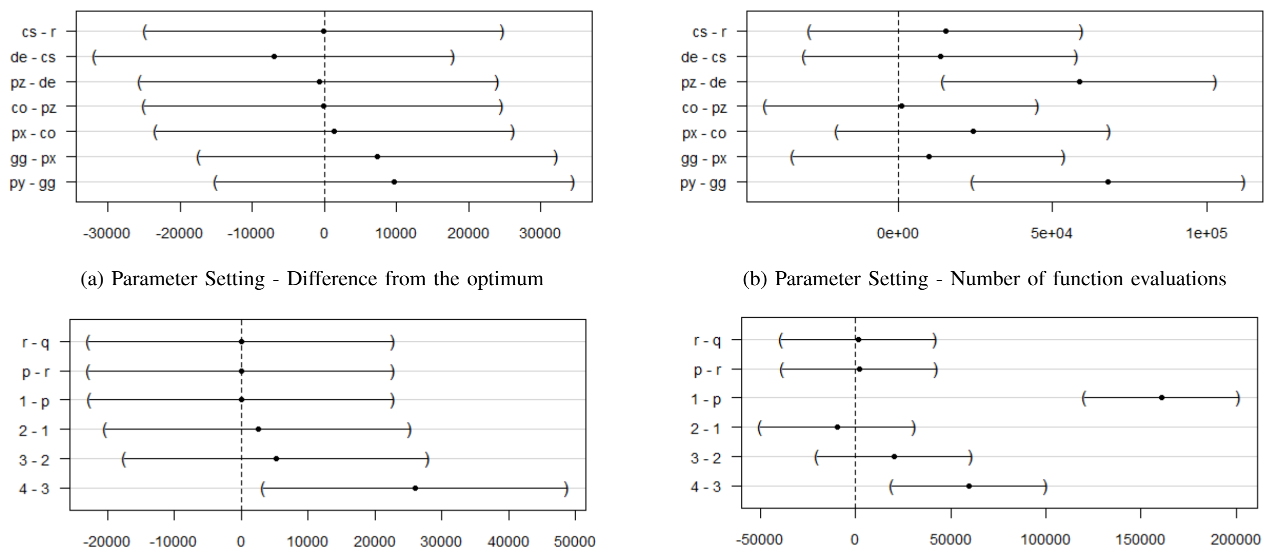
For the evaluation of the described methods the Black Box Optimization Benchmark (BBOB 2012)[3] has been used. It has 24 non-constrained non-linear functions[4] with real variables. The optimum value is known for all of them.

Due to the stochastic nature of the methods, each different combination had 15 independent executions in each one of the 24 problems with 5, 10 and 20 dimensions. Two stopping criteria have been used[5]: (i) number of function evaluations higher than 10^6 and (ii) $f_b - f_{opt} < 10^{-8}$. Where f_b is the function value of the best individual in the population and f_{opt} is the known optimal value. For each run, the following measures were collected:

[3]Available at http://coco.gforge.inria.fr/doku.php?id=bbob-2012

[4]For a complete description of the functions, see [39], available at http://coco.lri.fr/BBOB-downloads/download11.05/bbobdocfunctions.pdf

[5]The complete set of results is available online at: http://goo.gl/mMZ1Pv



(a) Parameter Setting - Difference from the optimum

(b) Parameter Setting - Number of function evaluations

(c) Variation Operators - Difference from the optimum

(d) Variation Operators - Number of function evaluations

Fig. 1. Post-hoc statistical analysis

- Number of function evaluations for convergence[6] (Nfeval)
- Difference from the optimum ($f_b - f_{opt}$)

For the statistical analysis, once there were violations from the standard Analysis of Variance premises, a modified model based in permutations [40] has been used. Due to thecomputational cost related to the permutation approach, the results of the 15 runs in each problem were collapsed in their average. This approach is justified statistically in [41]. For all analysis a significance level $\alpha = 0.05$ was considered. For the cases in which a significant effect was found ($p < 0.05$), analysis post-hoc through multiple comparison of means with sequential contrast have been made[7].

VII. Results

Figure 1 depicts the 95% confidence intervals of the post-hoc analysis. There, the methods are sorted by their mean and the hypothesis test verifies if the difference between the current method and its consecutive is 0.

The statistical analysis results reveal that with regards to the approaches for parameter setting no significant difference was found for the difference from the optimum. Regarding the number of function evaluations for convergence three groups can be made. The group which has presented the best results is composed by **cs**, **r** and **de**. The intermediary group is composed by **co**, **pz**, **px** and **gg** and in the third group with the worst results we have **py**. The was no statistical difference within groups.

These results indicate that, as expected, a pure exploratory strategy (**py**) decreases the speed of convergence. In addition to that, it seems that the use methods which keep a broader pool of parameters, such as **cs**, **r** and **de** are more beneficial to the DE performance.

Regarding the variation operators, there was no significant difference among the methods with multiple operators, which are, **q**, **p** and **r**. It is also important to highlight that in overall they have presented significantly better convergence speed than the methods using only one variation operator (mutation + crossover). Among the methods with only one operator, just the one which have utilized the operator **4** presented significantly worse performance in both Nfeval and $f_b - f_{opt}$.

Figure 2 shows the overall results with the respective confidence intervals and the interactions among parameter setting methods and methods for the configuration of operators. Each square has a different parameter setting approach and the x-axis shows the different operator settings. It can be seen that the use of multiple operators is beneficial regardless of the methods for operator configuration or parameter setting. Despite **q** and **p** present a "learning" capability, it seems that the solely knowledge about improvement is not enough to guide the algorithm towards better performance (for the tested problems). The results indicate that the only benefit of applying these approaches is the use of multiple operators.

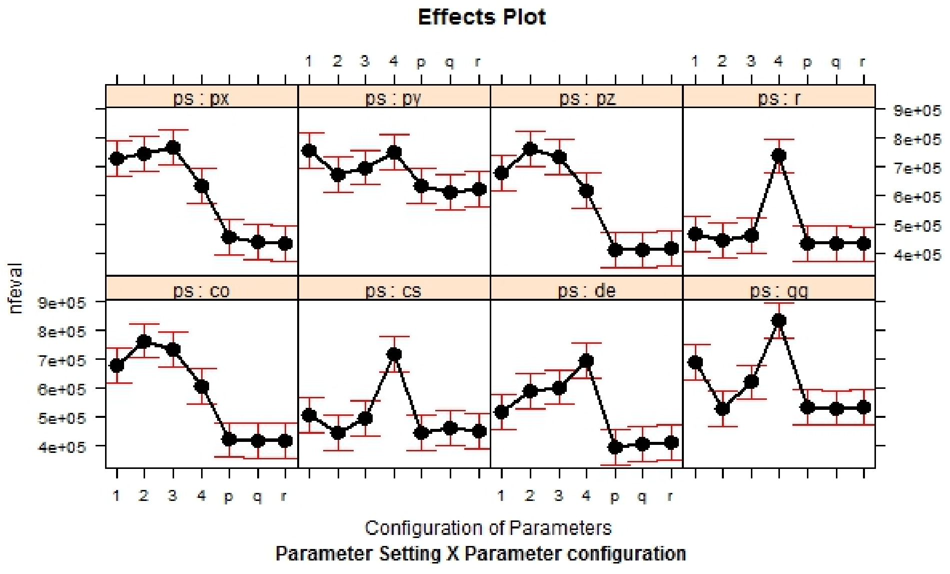
VIII. Conclusion and Future Work

This paper has presented a comprehensive study about Self-configuration in Differential Evolution Algorithm. A broad set of approaches were implemented and compared in a rigorous empirical experiment. For the tested benchmark the results reveal that the simplest approaches (pure randomization) present the best results when one considers the implementation complexity.

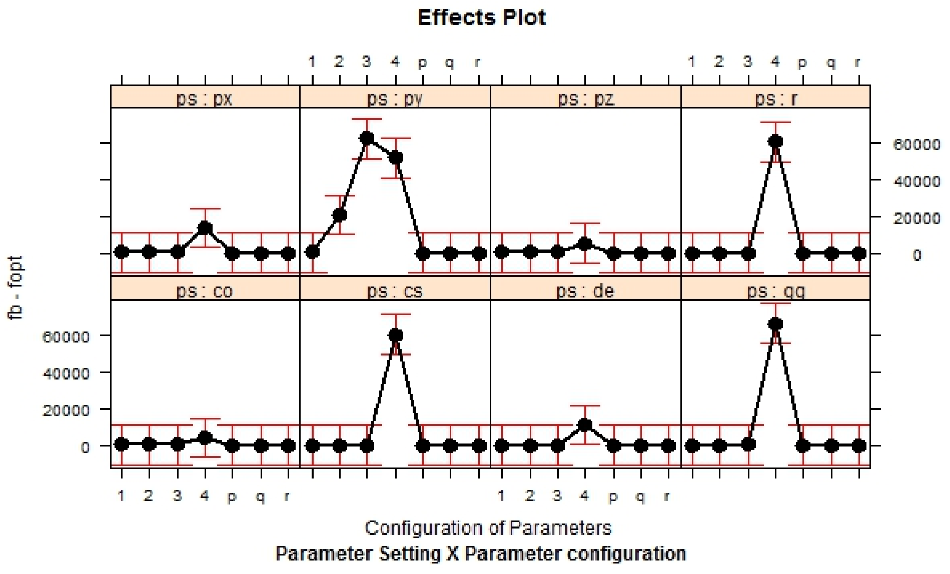
Although the methods with "learning" capability presented performance just as good as pure randomization, they can still prove their value in other applications. Stochastic Optimization, in which the objective function is stochastic and Online Optimization, in which the objective function changes

[6]The convergence criteria is met if $f_b - f_{opt} < 10^{-8}$. If the method does not converge the number of function evaluations is set to 10^6.

[7]Function **glht()** from R statistical software



(a) Number of Function Evaluations



(b) Difference from the Optimum

Fig. 2. Post-hoc analysis - Interactions

with respect to time, seem to be more interesting problems for this kind of approach. Problems with a higher number of design variables may also provide different insight about these methodologies once they would give more time for the learning process.

As future work, the authors intend to expand these experiments to discrete and constrained problems and include other reinforcement learning methods for the problem of operator selection. Some possible alternatives to the Q-learning are: the Upper Confidence Bounds, the Adaptive Pursuit, the ϵ-greedy and the Softmax algorithm.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces," Tech. Rep., 1995.

[2] M. Weber, V. Tirronen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Computing*, vol. 14, no. 11, pp. 1187–1207, 2010.

[3] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, ser. Natural Computing Series. Springer-Verlag, 2005.

[4] G. C. Onwubolu and D. Davendra, "Scheduling flow shops using differential evolution algorithm," *European Journal of Operational Research*, vol. 171, no. 2, pp. 674–692, 2006.

[5] F. Guimares, R. Silva, R. Prado, O. Neto, and D. Davendra, "Flow shop scheduling using a general approach for differential evolution," in *Handbook of Optimization*, ser. Intelligent Systems Reference Library, I. Zelinka, V. Snel, and A. Abraham, Eds. Springer Berlin Heidelberg, 2013, vol. 38, pp. 597–614.

[6] R. Lopes, A. Freitas, R. Pedosa Silva, and F. Guimares, "Differential evolution and perceptron decision trees for classification tasks," in *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, ser. Lecture Notes in Computer Science, H. Yin, J. A. Costa, and G. Barreto, Eds. Springer Berlin Heidelberg, 2012, vol. 7435, pp.

550–557.

[7] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[8] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A comparative study of differential evolution variants for global optimization," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2006, pp. 485–492.

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

[10] S. Das and A. Konar, "An improved differential evolution scheme for noisy optimization problems," in *Proceedings of the First international conference on Pattern Recognition and Machine Intelligence*, ser. PReMI'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 417–421.

[11] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[12] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Comput.*, vol. 10, no. 8, pp. 673–686, 2006.

[13] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO 2008)*. ACM, 2008, pp. 913–920.

[14] W. Gong, A. Fialho, and Z. Cai, "Adaptive strategy selection in differential evolution," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, ser. GECCO '10. New York, NY, USA: ACM, 2010, pp. 409–416.

[15] R. C. Pedrosa Silva, R. A. Lopes, and F. G. Guimarães, "Self-adaptive mutation in the differential evolution," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*. New York, NY, USA: ACM, 2011, pp. 1939–1946.

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

[17] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Congress on Evolutionary Computation'05*, 2005, pp. 1785–1791.

[18] J. Zhang and A. C. Sanderson, "Jade: Self-adaptive differential evolution with fast and reliable convergence performance." in *IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 2251–2258.

[19] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evolutionary Computation*, pp. 55–66, 2011.

[20] J. Brest, B. Boskovic, S. Greiner, V. Zumer, and M. S. Maucec, "Performance comparison of self-adaptive and adaptive differential evolution algorithms," *Soft Computing*, vol. 11, pp. 617–629, 2007.

[21] X. Liu, L. Shi, and R. Chen, "Jdel: Differential evolution with local search mechanism for high-dimensional optimization problems," in *Information and Automation*, ser. Communications in Computer and Information Science, L. Qi, Ed. Springer Berlin Heidelberg, 2011, vol. 86, pp. 347–352.

[22] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Differential Evolution with Self-Adaptation and Local Search for Constrained Multiobjective Optimization," in *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*. Trondheim, Norway: IEEE Service Center, May 2009, pp. 195–202.

[23] C.-H. Sun, C.-C. Chiu, M.-H. Ho, and C.-L. Li, "Comparison of dynamic differential evolution and self-adaptive dynamic differential evolution for buried metallic cylinder," *Research in Nondestructive Evaluation*, vol. 24, no. 1, pp. 35–50, 2013. [Online]. Available: http://dx.doi.org/10.1080/09349847.2012.699607

[24] D. Zaharie, "Critical values for the control parameters of differential evolution algorithm," in *Proceedings of MENDEL 2002, 8th International Mendel Conference on Soft Computing, June 5.7.2002, Brno, Czech Republic*, R. Matouek and P. Omera, Eds., 2002.

[25] J. Zhang and A. C. Sanderson, "Jade: adaptive differential evolution with optional external archive," *Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[26] D. Thierens, "An adaptive pursuit strategy for allocating operator probabilities," *Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05*, p. 1539, 2005.

[27] A. Fialho, M. Schoenauer, and M. Sebag, "Analysis of adaptive operator selection techniques on the royal road and long k-path problems," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 779–786.

[28] J. Maturana and F. Saubion, "A compass to guide genetic algorithms," in *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*. Springer-Verlag, 2008, pp. 256–265.

[29] A. Fialho, R. Ros, M. Schoenauer, and M. Sebag, "Comparison-based adaptive strategy selection with bandits in differential evolution," in *Proceedings of the 11th international conference on Parallel problem solving from nature: Part I*, ser. PPSN'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 194–203.

[30] K. Li, A. Fialho, and S. Kwong, "Multi-objective differential evolution with adaptive control of parameters and operators," in *Proceedings of the 5th international conference on Learning and Intelligent Optimization*, ser. LION'05. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 473–487.

[31] W. Gong, A. Fialho, Z. Cai, and H. Li, "Adaptive strategy selection in differential evolution for numerical optimization: An empirical study," *Inf. Sci.*, vol. 181, no. 24, pp. 5364–5386, 2011.

[32] A. Fialho, L. Costa, M. Schoenauer, and M. Sebag, "Extreme value based adaptive operator selection," in *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 175–184.

[33] J. M. Whitacre, T. Q. Pham, and R. A. Sarker, "Use of statistical outlier detection method in adaptive evolutionary algorithms," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ser. GECCO '06. New York, NY, USA: ACM, 2006, pp. 1345–1352.

[34] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[35] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, 1989.

[36] M. Wunder, M. L. Littman, and M. Babes, "Classes of multiagent q-learning dynamics with epsilon-greedy exploration," in *ICML*, 2010, pp. 1167–1174.

[37] M. Guo, Y. Liu, and J. Malec, "A new q-learning algorithm based on the metropolis criterion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, pp. 2140–2143, 2004.

[38] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '08. New York, NY, USA: ACM, 2008, pp. 913–920. [Online]. Available: http://doi.acm.org/10.1145/1389095.1389272

[39] S. Finck, N. Hanseny, R. Raymond, and A. Augerx, "Real-parameter black-box optimization benchmarking 2010: Presentation of the noiseless functions," *Working Paper 2009/20, compiled March 24, 2012*, pp. 1–102, 2010.

[40] M. J. Anderson and J. Robinson, "Permutation tests for linear models," *Australian & New Zealand Journal of Statistics*, vol. 43, no. 1, pp. 75–88, 2001.

[41] D. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons, 2008.