2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

An Analysis of the Automatic Adaptation of the Crossover Rate in Differential Evolution

Carlos Segura*, Carlos A. Coello Coello†, Eduardo Segredo‡, Coromoto León‡

*UMI LAFMIA 3175 CNRS at CINVESTAV-IPN, Departamento de Computación,
Av. IPN No. 2508, Col. San Pedro Zacatenco, México D.F. 07300, MÉXICO
† CINVESTAV-IPN, Departamento de Computación (Evolutionary Computation Group),
Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F. 07300, MÉXICO
‡Parallel Algorithms and Languages Group, Departamento de Estadística, Investigación
Operativa y Computación, Facultad de Matemáticas, Universidad de La Laguna (ULL).
Avda. Astrofísico Fco. Sánchez s/n. 38071 La Laguna, S/C de Tenerife, Spain
Email: csegura@ull.es, ccoello@cs.cinvestav.mx, esegredo@ull.es, cleon@ull.es

***Abstract*—Differential Evolution (DE) is a very efficient metaheuristic for optimization over continuous spaces which has gained much popularity in recent years. Several parameter control strategies have been proposed to automatically adapt its internal parameters. The most advanced DE variants take into account the feedback obtained in the optimization process to guide the dynamic setting of the DE parameters. Indeed, the automatic adaptation of the crossover rate (CR) has attracted a lot of research in the last decades. In most of such strategies, the quality of using a given CR value is measured by considering the probability of performing a replacement in the DE selection stage when such a value is applied. One of the main contributions of this paper is to experimentally show that the probability of replacement induced by the application of a given CR value and the quality of the obtained results are not as correlated as expected. This might cause a performance deterioration that avoids the achievement of good quality solutions even in the long-term. In addition, the experimental evaluation developed with a set of optimization problems of varying complexities clarifies some of the advantages and drawbacks of the different tested strategies. The only component varied among the different tested schemes has been the CR control strategy. The study presented in this paper provides advances in the understanding of the inner working of several state-of-the-art adaptive DE variants.**

I. Introduction

Differential Evolution [1] (DE) is a very popular metaheuristic specially suited for real-parameter optimization. From its inception, it has been consistently ranked as one of the most efficient metaheuristics for solving continuous optimization problems. It has been successfully applied both with benchmark problems and with complex optimization problems arising in practical applications [2]. Its success was initially demonstrated at the *First International Contest on Evolutionary Computation* [3]. Subsequently, other DE variants also performed adequately in several contests, such as in the *2005 IEEE Congress on Evolutionary Computation* (CEC) competition on real parameter optimization [4], or in the special issue on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems, recently organized for the *Soft Computing* journal [5], [6]. Some of the domains where DE has provided competitive results are control systems [7] and electrical power systems [8]. The reader is referred to [9] for a historical overview of DE.

The amount of research conducted in DE in the last decades is huge. For instance, DE has been adapted to face multi-objective [10], constrained [11] and large-scale [12] optimization problems. One of the topics that have attracted a large amount of research concerns the proper parameterization of DE. Initially, DE was presented as a robust scheme, with a low number of easily tunable parameters [1]. However, subsequent studies showed that DE is very sensitive to the setting of the control parameters [13], [14], so generally, the parameter setting stage is crucial to successfully apply DE. Furthermore, several DE trial vector generation strategies have been proposed [15], hampering the choice of DE parameters and components.

The problem of properly setting the parameters is not specific of DE. In fact, this is one of the persisting grand challenges for Evolutionary Computation (EC) [16]. Parameter setting strategies are commonly divided into two categories: parameter *tuning* and parameter *control*. In parameter *tuning* the objective is to identify the best set of values for the parameters of a given Evolutionary Algorithm (EA), and then, such an EA is executed using these values which remain fixed during the complete run. On the contrary, the aim of parameter *control* is to design control strategies that select the most suitable values for the parameters at each stage of the search process, while the algorithm is executed. It has been empirically and theoretically shown that different values of the parameters might be optimal at different stages of the optimization process [17], [18]. In such cases, it seems more appropriate to apply parameter control strategies that enable the parameter values to adapt or change during the course of an EA run.

In the case of DE, several schemes that consider the simultaneous use of different DE components and parameter values have been devised. For instance, in [19] two DE schemes with random and varying parameters were proposed, while in [20] three trial vector generation strategies were simultaneously considered. In addition, some adaptive schemes have been developed [21], [22] that take into account the feedback obtained in the own optimization process in order to set the

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

DE parameters dynamically. A large amount of research has focused on the adaptation of the crossover rate (CR). The most popular parameter control strategies grant more resources to those CR values that maximize the number of replacements carried out in the selection phase of DE. Since the selection phase of DE is elitist, the principle behind such strategies is that a larger amount of replacements induce the achievement of solutions of higher quality. However, to the best of our knowledge such a correlation has not been ever studied. It is also interesting to mention that in most up-to-date DE variants, not only CR is adapted, but other components are also modified. Thus, most of the comparisons currently published in the specialized literature do not measure the benefits obtained by the single adaptation of CR.

In this research, an empirical analysis of the correlation between the quality of the obtained solutions and the probability of replacement induced by different CR values is carried out. Surprisingly, it shows that the correlation is not as clear as expected, meaning that the performance of the schemes based on such correlation can severely deteriorate in several optimization problems. In fact, the experimental evaluation shows that all the tested schemes fail to optimize some of the considered benchmark problems even in the long-term. The analyses also show that the reasons of the deterioration of the performance differ among the tested adaptive DE variants. In spite of that, the experimental validation developed in this research shows that, in most cases, using such adaptive schemes is preferable than using random strategies to set the crossover rate.

The rest of the paper is organized as follows. Section II presents the fundamentals of DE. A summary of some of the most popular state-of-the-art adaptive DE variants is given in Section III. Then, our experimental validation is presented in Section IV. Finally, conclusions and some lines of future work are given in Section V.

II. Fundamentals of Differential Evolution

DE was originally devised as a search strategy for single-objective continuous optimization problems [1]. In continuous optimization, the variables governing the system to be optimized are given by a vector $\vec{X} = [x_1, x_2, x_3, ..., x_D]$, where each variable x_i is a real number. The number of variables (D) defines the dimensionality of the optimization problem. Finally, the objective function $f(\vec{x})(f : \Omega \subseteq \mathbb{R}^D \rightarrow \mathbb{R})$ measures the quality of each set of variables. The purpose of the optimization — considering a minimization problem — is to find a vector $\vec{x*} \in \Omega$ in which $f(\vec{x*}) \leq f(\vec{x})$ holds for all $\vec{x} \in \Omega$. The problems most typically addressed with DE are box-constrained optimization problems. In these cases, the region Ω is specified by indicating a lower (a_j) and upper (b_j) bound for each variable in the problem. Thus, the feasible region can be described as $\Omega = \prod_{j=1}^{D}[a_j, b_j]$.

DE is a population-based stochastic algorithm that belongs to the broad class of Evolutionary Algorithms (EAs). As with other EAs, it randomly initializes a population (P) with NP individuals or vectors ($P = \{\vec{X}_1, ..., \vec{X}_{NP}\}$). Each individual is a vector with D real numbers. The value of the j^{th} variable of individual X_i is denoted by $X_{i,j}$. Then, the population evolves over successive iterations to explore the search space.

The DE vector generation strategy is used to create new vectors based on the contents of the current population. Several vector generation strategies have been proposed [15]. The strategies proposed share the common property of taking into account the differences among the vectors present in the current population to create new vectors. The strategy applied in this research is described in this section. The reader is referred to [2], [9] for a more detailed description of other DE variants.

At each DE iteration, the following steps are executed. First, for each vector in the population — called *target vector* — a new *mutant vector* is created using a vector generation strategy. Then, the mutant vector is combined with the target vector to generate the *trial vector*. After generating NP trial vectors, each one is compared against its corresponding target vector. In each comparison the best one is selected to survive. In case of a tie, the survivor depends on the implementation. In our case, the new generated trial vector survives.

In the vector generation strategy, the term *base vector* denotes an initial vector that is subsequently perturbed to generate the mutant vector. The perturbation is done by considering one or several differences among other vectors in the population. In order to classify the different variants of DE, the notation DE/x/y/z was introduced in [1]. The term x specifies how to select the base vector. The term y is the number of difference vectors used. Finally, z denotes the crossover or combination scheme. Thus, x and y set up the mutation strategy, and z the crossover scheme.

In this research the DE/rand/1/bin variant has been considered. Note that this scheme has attained competitive results with respect to other variants [23]. In the "rand" strategy, any vector in the population different from the target vector is selected as the base vector. Thus, the mutant vector (V_i) for target vector i is created with Eq. (1). In this equation, r_1, r_2 and r_3 are mutually exclusive integers chosen at random from the range [1, NP]. In addition, they are all different from the index i. The mutation scale factor F is a control parameter for scaling the difference vector.

$$\vec{V}_i = \vec{X}_{r1} + F \times (\vec{X}_{r2} - \vec{X}_{r3}) \tag{1}$$

The crossover strategy applied in this research is the binomial (bin) one. As in other EAs, the crossover operator is controlled by means of the crossover rate (CR). In the bin strategy, the trial vector (U_i) is generated using Eq. (2). $rand_{i,j}$ is a uniformly distributed random number in the range [0,1], and $j_{rand} \in \{1, 2, ..., D\}$ is a randomly chosen index which ensures that at least one variable is taken from the mutant vector. For the remaining cases, we see that the probability of the variable being inherited from the mutant is CR. Otherwise, the variable of the target vector is considered.

$$U_{i,j} = \begin{cases} V_{i,j} & if\ (rand_{i,j} \leq CR\ or\ j = j_{rand}) \\ X_{i,j} & otherwise \end{cases} \tag{2}$$

The vector generation strategy, as described above, might generate trial vectors outside the feasible region. In the case of box-constrained optimization problems — which are the ones

faced in this research — several ways of dealing with this scenario have been proposed [15]. First, the brick wall penalty scheme sets the offending vector's objective function value high enough to guarantee that it will not be selected. Since it does not take into account the particular definition of Ω, this scheme is, generally, not very appropriate. The bounce-back method generates a new value that lies between the value of the base variable and the bound being violated. Finally, another widely used scheme is based on randomly reinitializing the offending values. As this last approach is the most unbiased and has yielded promising results, it is the one we have used in this paper.

III. Adaptive DE Variants

Since the earliest years of DE, there has been a significant amount of research focused on the proper parameterization of DE. Specifically, the parameterization of CR and F have been widely studied. The initial experimentation carried out in [1] showed that "DE requires few control variables, is robust and easy to use". However, experimentation with additional benchmark problems uncovered the sensitivity of DE's control parameters [13], [14]. Although some general recommendations have been given concerning the proper choice of CR and F, the most up-to-date DE variants consider the automatic adaptation of such parameters.

The most popular DE variants consider the simultaneous adaptation of several components and/or parameters. For instance, in DESAP [24], F, CR and NP are adapted, while jDE [25] only considers the adaptation of CR and F. Some comparative analyses including several of the most popular schemes have been provided in the last years [26], [9]. However, since most of them differ in several components, it is not easy to predict the benefits attained by each single modification. Particularly, there have been some controversies concerning the benefits obtained by adapting CR and F [27], so it is interesting to study each single adaptation independently from each other. The single adaptation of F is analyzed in [23]. The current research focuses on the single adaptation of CR.

In this section, some of the most popular adaptive DE variants are presented. Specifically, considering the results obtained in [26], the strategies jDE [25], competitive DE (cDE) [28], JADE [22] and SaDE [29] have been selected. It is important to mention that all of them try to grant more resources to the CR values that produce a larger number of replacements in the selection stage of DE. Thus, all of them assume that a larger number of replacements provides solutions with a higher quality. In the literature there are other (not so popular) adaptive schemes that are not based in such an assumption. For instance, in [30] the increase of fitness obtained in each replacement is considered to perform the adaptation, while in [31] the diversity is also taken into account. The analysis of such kind of schemes is beyond the scope of this paper.

A. *j*DE

jDE was presented by Brest et al. [25] as a simple and efficient adaptive DE variant that self-adapts F and CR. A pair containing a value for F and CR is encoded within each individual of the population. Initially, F and CR are set randomly for each individual of the population. Each time a new individual is created, the CR and F values can be acquired from the target vector or are randomly generated. Specifically, they are randomly generated with probabilities τ_1 and τ_2, respectively. New values of F are generated in the range $[F_{min}, F_{max}]$, while new values of CR are generated in the range $[CR_{min}, CR_{max}]$.

B. *c*DE

cDE was presented by Tvrdík [28] as an adaptive scheme where several trial vector generation strategies are stored in a pool and compete for getting resources. The strategies stored in the pool might differ in any component and/or parameter, so this can be viewed as a general adaptive scheme. The pool contains H strategies, and each time that a new individual is created, a strategy is randomly selected with probability $q_h, h = 1, 2, ...H$. Specifically, the probability q_h is given by Eq. (3). In such equation, n_h is the number of replacements that has been carried out when the h setting has been used, and $n_0 > 0$ is an input parameter. The aim of n_0 is to prevent drastic changes in the probabilities between consecutive generations. Note that considering Eq. (3), the initial probabilities are set uniformly, i.e., $q_h = \frac{1}{H}$. In order to avoid degeneration of the search process, if any probability decreases below some given limit δ, the memory of the scheme is erased, i.e., the probabilities are reset to their initial values. Experimentation with this adaptive variant has shown that considering a large number of strategies in the pool is not adequate [26].

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H} (n_j + n_0)} \tag{3}$$

C. JADE

JADE was presented by Zhang and Sanderson [22] as a novel DE variant that includes the adaptation of CR and F. It also includes other algorithmic modifications, such as the use of a novel trial vector generation strategy and the incorporation of an archive. Since in this work we are mainly interested in the adaptation scheme, the other details of this approach are omitted.

Adaptation of CR and F is carried out as follows. Each time an individual must be created, new CR and F values are generated. The CR value is generated according to a normal distribution of mean μ_{CR} and standard deviation 0.1, and then truncated to $[0, 1]$. The F value is generated according to a Cauchy distribution with location factor μ_F and scale parameter 0.1. If the generated value is lower than 0, it is regenerated. If it is higher than 1, it is truncated to 1. JADE adapts μ_{CR} and μ_F considering the feedback obtained during the optimization process. Specifically, in each generation t each CR and F values that generate an individual that replaces its corresponding target vector, are stored in the pools S_{CR_t} and S_{F_t}. At the end of each generation, μ_{CR} is updated with Eq. (4) and μ_F is updated with Eq. (5), where $mean_A$ is the arithmetic mean and $mean_L$ is the Lehmer mean. The parameter c is an input parameter and represents the factor's adaptation speed.

$$\mu_{CR_{t+1}} = (1 - c) \times \mu_{CR_t} + c \times mean_A(S_{CR_t}) \tag{4}$$

$$\mu_{F_{t+1}} = (1 - c) \times \mu_{F_t} + c \times mean_L(S_{F_t}) \quad (5)$$

D. SaDE

SaDE was presented by Qin and Suganthan [32], [29] as a self-adaptive scheme with strategy adaptation. A set of mutation strategies is stored in a pool, and after a learning period (LP generations), each strategy k is used in the creation of a new individual with a probability p_k. The value p_k is proportional to the probability of performing a replacement with such strategy in previous generations. The pool does not include the value of CR and/or F. Instead, the F values are randomly generated from a normal distribution with mean 0.5 and standard deviation 0.3. The CR values are generated from a normal distribution with mean CR_m and standard deviation 0.1. If the generated value is lower than 0 or higher than 1, it is regenerated. Initially, CR_m is set to 0.5. Successful CR values, i.e., those that produce a trial vector entering the next generation, are stored in a queue (Q_{CR}) with a maximum size Mem_{CR}. After the learning period, CR_m is updated to the median of Q_{CR} in each generation.

IV. Experimental Study

This section shows the experimental study developed to analyze the benefits and drawbacks of the aforementioned CR control strategies. The tested CR control strategies have been the ones defined in jDE, cDE, JADE and SaDE. They have been incorporated in a DE/rand/1/bin strategy. Following the recommendations given in [23], NP was set to 50, and F was generated following a Cauchy distribution with location factor 0.5 and scale parameter 0.1. Table IV shows the parameterization of each control strategy. They have been set following the recommendations given in [26].

The analyses were performed using the benchmark problems described in [33], which are a set of 19 scalable continuous optimization problems to be minimized. The parameter *D* allows setting the number of variables in the problems. In our study, it was set to 50. In every experiment, each execution was repeated 1,000 times.

Since stochastic algorithms were considered, comparisons were carried out by applying the following statistical analysis, assuming a significance level of 5%. First, a *Shapiro-Wilk test* was performed to check whether or not the values of the results followed a Gaussian distribution. If so, the *Levene test* was used to check the homogeneity of the variances. If the samples had equal variance, an ANOVA *test* was done; if not, a *Welch test* was performed. For non-Gaussian distributions, the non-parametric *Kruskal-Wallis* test was used. In this work, the sentence "algorithm A is better than algorithm B" means that the differences between them are statistically significant, and that the mean and median obtained by A are lower—one of the metrics might be equal—than the mean and median achieved by B.

A. Correlation between quality and replacement probability

The tested CR control strategies grant more resources to those CR values that have been successfully applied in previous generations, i.e., those that have created trial vectors which have survived to the following generations. This is a greedy approach based on the principle that performing many replacements will result in a better quality. However, there is no a mathematical proof of such a relation. For instance, some CR values might perform a lower number of replacements, but in each replacement the improvement might be larger, resulting in better final solutions. This first experiment studies the correlation between final quality and replacement probability with some of the most widely used benchmark problems. A DE/rand/1/bin strategy was executed considering several fixed CR values. Specifically, 51 CR values uniformly distributed in the range $[0, 1]$ were considered. The stopping criterion was set to 250,000 function evaluations.

TABLE I. Parameterization of the CR control strategies

jDE	CR_{min} = 0, CR_{max} = 1, τ_1 = 0.1
cDE	$H = 6$, CRs equidistributed in [0,1], $n_0 = 2$, $\delta = \frac{1}{5 \times H}$
JADE	$c = 0.1$
SaDE	$LP = 50$, $Mem_{CR} = 50$

Figure 1 shows, for nine of the problems, the mean of the error obtained at the end of the executions, as well as the mean number of replacements performed for each CR value. The problems are the ones where larger final errors were obtained. We can appreciate that, in the general case, there is no a clear correlation between the number of replacements and the quality. For instance, in F2, the number of replacements does not seem to depend on CR, while the quality highly depends on CR. In addition, in most cases, the largest CR values produce a very large number of replacements. However, this large number of replacements does not result in a better quality. The reason is that when very large CR values are used, the crossover rejects most of the values of the target vector. This might result in a quick convergence, so intensification might be promoted. However, in such cases, diversity is quickly lost, so the improvements obtained in each replacement might be negligible.

The nonexistence of a clear correlation means that the reasons of the relative success of adaptation of CR are more complex than expected. Note that in all the tested algorithms, the randomness introduced by the control strategy is very large. For instance, jDE selects several CR values uniformly, regardless of the feedback obtained in the optimization process, while in JADE, instead of adjusting the value of CR, a random distribution to generate CR values is adjusted. Thus, it seems that part of the success is due to the increase of randomness attained with the control strategies. In fact, a more traditional self-adaptive scheme that does not introduce so much randomness has not been as successful as the ones analyzed in this paper [34]. Another evidence that the inclusion of randomness might help is the study presented in [35]. In such a study it was shown that DE might be deceived into converging on the wrong peak due to the lower amount of diversity introduced by DE with respect to other metaheuristics. Anyway, the performed experiments show that most of the adaptive schemes are assuming a relation that generally does not hold, and as we will experimentally demonstrate, this might provoke incorrect decisions that might result in large performance deterioration.

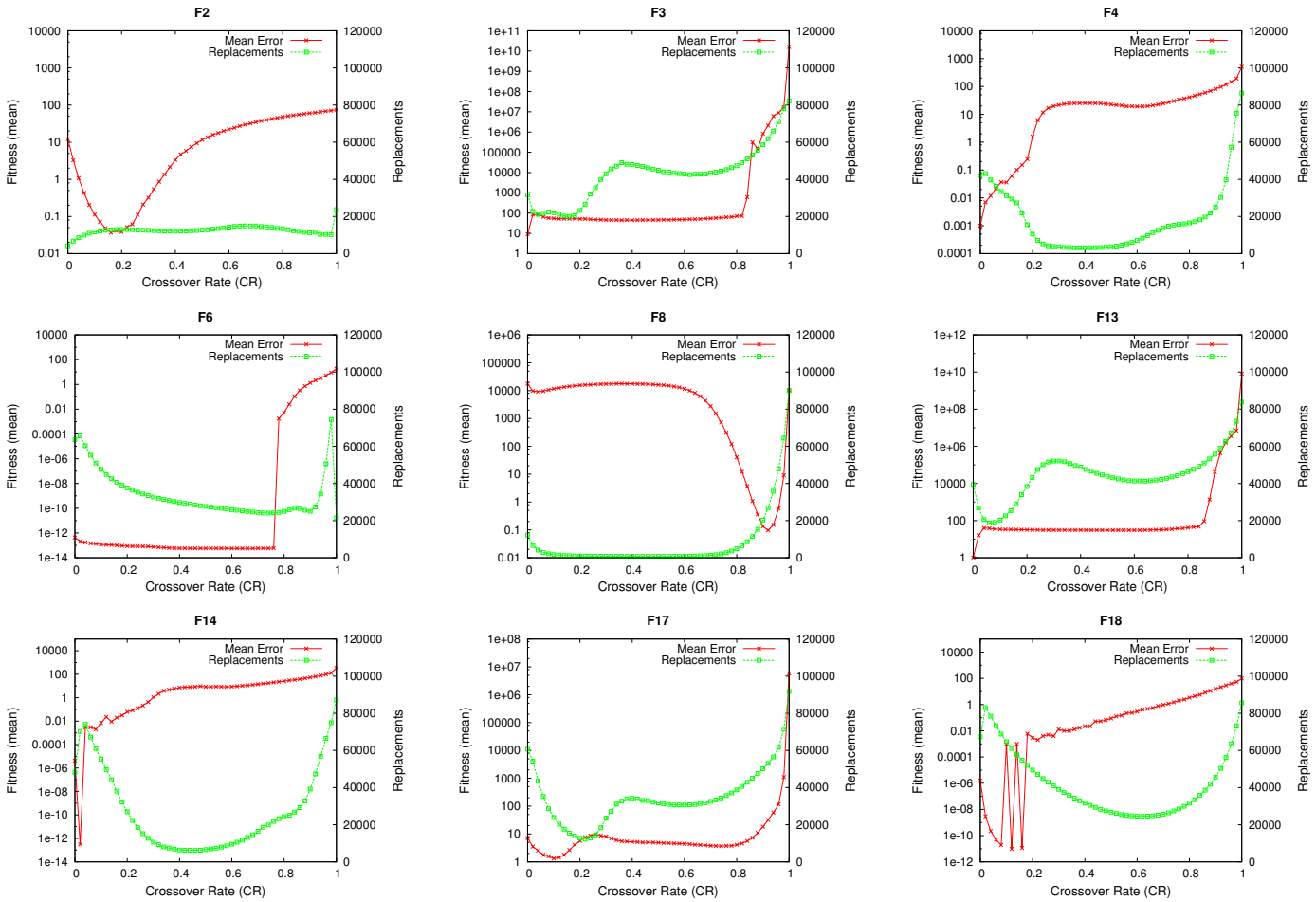

Fig. 1. Relation between the quality and the number of replacement performed in DE

B. *Performance of* CR *control strategies*

The aim of this experiment has been to analyze the performance of the different CR control strategies, by comparing the results obtained by them with the ones obtained with fixed CR values. Besides the aforementioned CR control strategy, a random uniform scheme was also considered. In such case, the CR values are generated randomly in the range $[0, 1]$. The same stopping criterion as in the previous experiment was considered. Each CR strategy was statistically compared against the 51 cases with fixed CR.

Table II shows, for each tested problem, the percentage of cases where DE with a given CR control strategy was superior ($\uparrow$) or inferior ($\downarrow$) than DE with a fixed CR value. It also shows the mean across all tested problems. We can appreciate that, in general, the adaptive strategies perform adequately being only worse than a small percentage of fixed configurations. In addition, when comparing the number of cases where the adaptive schemes has been worse than the fixed schemes, the inferiority of the uniform scheme is clear, meaning that the feedback is providing a good orientation towards the achievement of high-quality solutions.

Considering the overall results, the jDE scheme achieved the best performance. However, in every strategy, there are several problems with a high percentage of fixed configurations that are better than the adaptive scheme. For instance, in every strategy there is at least one problem where the number of fixed configurations that are better than it is larger than 50%. This means that the CR control strategies are not as robust as expected. A deeper analysis of the reasons why this happens is developed in the following section.

C. *Analysis of the Long-term Behavior*

In the previous analysis we showed that there are several problems where the adaptive strategies fail to reach solutions as good as those achieved by DE with the best fixed CR values. However, finding such proper CR values is a time-consuming task, so it is interesting to show whether the adaptive schemes are able to reach such solutions in the long-term. Specifically, the same adaptive schemes as in the previous experiment were executed but the stopping criterion was now set to 2,500,000 function evaluations. Each model was compared against DE with fixed CR values, but considering 250,000 function evaluations. Table III shows the results of the statistical comparison. We can appreciate that even if very long executions are run, adaptive schemes are not able to attain the quality obtained by the best fixed CR values in some problems. Moreover, considering the overall results, the adaptation of JADE seems to be the most robust, so the most adequate adaptation depends on the considered stopping criterion.

TABLE II. STATISTICAL COMPARISON BETWEEN ADAPTIVE AND FIXED SCHEMES (250,000 EVALUATIONS)

	jDE		cDE		JADE		SaDE		Uniform	
	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓
F1	7.84	0	7.84	0	7.84	0	7.84	0	7.84	0
F2	47.05	52.94	39.21	60.78	72.54	15.68	66.66	19.60	43.13	54.90
F3	72.54	1.96	37.25	37.25	19.60	78.43	27.45	60.78	54.90	19.60
F4	86.27	13.72	88.23	5.88	92.15	3.92	86.27	7.84	76.47	21.56
F5	37.25	1.96	37.25	1.96	47.05	0	47.05	0	37.25	43.13
F6	54.90	0	54.90	0	54.90	0	54.90	0	54.90	0
F7	37.25	0	37.25	0	37.25	0	37.25	0	35.29	0
F8	90.19	9.80	90.19	7.84	54.90	41.17	64.70	33.33	82.35	17.64
F9	50.98	0	31.37	41.17	50.98	0	50.98	0	43.13	49.01
F10	43.13	45.09	52.94	0	54.90	0	54.90	0	41.17	52.94
F11	58.82	0	31.37	45.09	58.82	0	58.82	0	45.09	43.13
F12	78.43	0	19.60	54.90	19.60	52.94	25.49	39.21	68.62	0
F13	96.07	3.92	76.47	3.92	23.52	56.86	25.49	56.86	84.31	3.92
F14	82.35	0	86.27	5.88	96.07	0	88.23	0	70.58	19.60
F15	33.33	0	33.33	0	33.33	0	33.33	0	33.33	0
F16	68.62	21.56	21.56	49.01	23.52	47.05	60.78	29.41	39.21	43.13
F17	82.35	13.72	58.82	35.29	86.27	9.80	94.11	0	70.58	15.68
F18	74.50	17.64	50.98	17.64	70.58	9.80	76.47	11.76	56.86	39.21
F19	41.17	0	49.01	0	49.01	0	49.01	0	39.21	49.01
Mean	60.15	9.58	47.56	19.29	50.13	16.60	53.13	13.62	51.80	24.87

TABLE III. STATISTICAL COMPARISON BETWEEN ADAPTIVE (2,500,000 EVALUATIONS) AND FIXED SCHEMES (250,000 EVALUATIONS)

	jDE		cDE		JADE		SaDE		Uniform	
	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓
F1	7.84	0	7.84	0	7.84	0	7.84	0	7.84	0
F2	47.05	52.94	39.21	60.78	76.47	0	72.54	0	43.13	54.90
F3	100	0	100	0	98.03	1.96	47.05	29.41	100	0
F4	86.27	13.72	88.23	5.88	92.15	3.92	86.27	7.84	76.47	21.56
F5	37.25	1.96	37.25	1.96	47.05	0	47.05	0	37.25	43.13
F6	100	0	54.90	0	100	0	100	0	100	0
F7	37.25	0	37.25	0	37.25	0	37.25	0	35.29	0
F8	100	0	100	0	72.54	27.45	74.50	23.52	100	0
F9	50.98	0	50.98	0	50.98	0	50.98	0	43.13	49.01
F10	43.13	45.09	52.94	0	54.90	0	54.90	0	41.17	52.94
F11	58.82	0	47.05	43.13	58.82	0	58.82	0	45.09	43.13
F12	100	0	100	0	100	0	100	0	100	0
F13	100	0	100	0	96.07	3.92	96.07	3.92	100	0
F14	82.35	0	90.19	0	96.07	0	88.23	0	70.58	0
F15	33.33	0	33.33	0	33.33	0	33.33	0	33.33	0
F16	100	0	49.01	0	100	0	100	0	100	0
F17	100	0	100	0	100	0	100	0	100	0
F18	78.43	0	82.35	0	100	0	78.43	0	56.86	0
F19	41.17	0	49.01	0	49.01	0	49.01	0	39.21	49.01
Mean	68.62	5.98	64.18	5.88	72.13	1.96	67.48	3.40	64.70	16.50

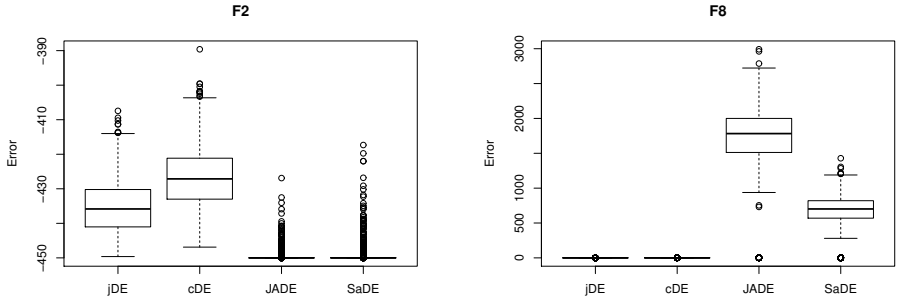

Fig. 2. Box-plots of the results obtained in 2,500,000 function evaluations

In order to have a better insight on the reasons of the performance deterioration, the problems F2 and F8 were selected. F2 was selected because jDE and cDE fail to optimize it, while F8 was selected because of the improper performance of JADE and SaDE. Figure 2 shows the boxplots of the obtained results. In F2, we can appreciate that jDE and cDE fail to reach the

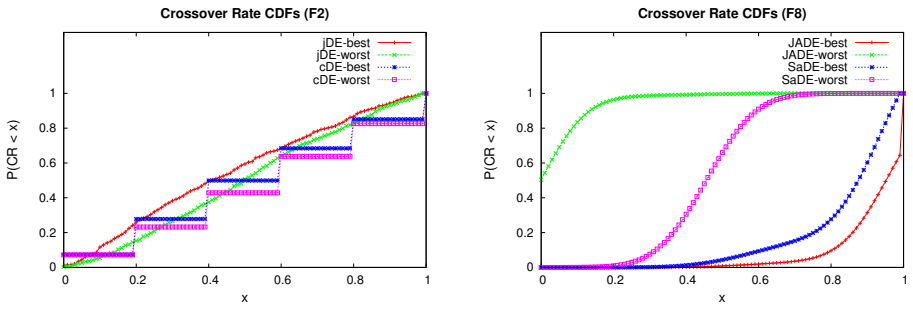

Fig. 3. Cumulative Distribution Functions of the CR generated by different adaptive schemes

optimal value (-450) in every execution, so they consistently reach non-optimal values. In F8 the behavior of JADE and SaDE is different. We can appreciate that in several cases, they are able to reach the optimal values. However, in most cases the optimization fails.

In order to better understand the differences among the schemes, the best and worst executions were identified. Figure 3 shows the cumulative distribution function (CDF) of the used CR values. In the case of F2, the CDFs are similar even when comparing the best with the worst executions. In fact, we can appreciate that both jDE and cDE tend to distribute the generated values along the whole valid range. For this reason, differences among executions are not so large. The main drawback is that large CR values might provoke a large drop in diversity. Therefore, for problems such as F2, where maintaining diversity is important, jDE and cDE fail to reach high-quality solutions. The CDFs for F8 shows that JADE and SaDE are more focused on specific CR regions. However, they are not always selecting the same region. By inspecting Figure 1, we know that high-quality results are obtained with large CR values. However, the number of replacements performed by different CR values is constant for medium CR values. Since, JADE and SaDE search for correct CR values by performing small changes to the mean of the Gaussian distribution, they might not reach the proper CR region.

V. Conclusions and Future Work

DE is one of the most popular population-based metaheuristics for continuous optimization. In recent years, DE has attracted a lot of attention because of its simplicity and efficiency. Particularly, several adaptive DE variants that take into account the feedback obtained in the optimization process itself in order to set the DE parameters in a dynamic way, have been devised. Most up-to-date DE variants adapt several parameters and/or components simultaneously and incorporate other algorithmic modifications, hampering the analysis of the benefits provided by each single adaptation. Particularly, several adaptive variants set the CR values dynamically, but in order to fully understand their inner working mechanisms, further studies are required. This paper provides a deep analysis of some of the most popular strategies used to adapt the CR values. The strategies tested in this study are based on the principle that the CR values that provoke a larger number of replacements in DE provide solutions with a higher quality. However, the analysis reported in this paper shows that there are several cases where this relation does not hold. This fact might provoke some improper decisions in such schemes which might result in large performance deterioration. The schemes tested in this research have been jDE, cDE, JADE and SaDE. Computational results show that these adaptive variants usually obtain competitive results, but at the same time, they show that no one is robust enough to successfully face the complete set of benchmark problems adopted here. Additionally, the computational study reveals that while jDE and cDE tends to use CR values distributed in the whole valid range, JADE and SaDE tends to focus on specific regions of the valid range. Thus, the variability of the results obtained by JADE and SaDE is usually larger. Another interesting conclusion is that the most effective adaptive variant depends on the stopping criterion established. This is due to the fact that the different variants provoke a different relationship between exploration and exploitation. For instance, while jDE and cDE can use large CR values since the beginning of the optimization process, JADE and SaDE start with medium CR values that are gradually modified during the execution. In some way, while jDE and cDE perform a global search over the CR values, JADE and SaDE perform a local search.

Since the study presented here has revealed that the correlation between quality and number of replacements does not generally hold, new adaptive variants might be proposed. In our opinion, it is very promising to test alternative schemes that are not exclusively based on granting more resources to those CR values that provide a larger number of replacements. For instance, the relationship between diversity and CR studied in [36] might be used to define new control strategies. Since the most adequate strategy has depended on the stopping criterion established, another interesting research line would be to include the stopping criterion as an input of the parameter control strategy.

Acknowledgement

This work was partially supported by the EC (FEDER) and the Spanish Ministry of Science and Innovation as part of the 'Plan Nacional de I+D+i', with contract number TIN2011-25448. The second author is also affiliated to the UMI LAFMIA 3175 CNRS at CINVESTAV-IPN. The work of Eduardo Segredo was funded by grant FPU-AP2009-0457.

References

[1] 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, Dec. 1997.

[2] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.

[3] R. Storn and K. Price, "Minimizing the real functions of the icec'96 contest by differential evolution," in *IEEE International Conference on Evolutionary Computation, 1996 (ICEC'96)*, 1996, pp. 842–844.

[4] A. K. Qin and P. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *The 2005 IEEE Congress on Evolutionary Computation (CEC'05)*, vol. 2, 2005, pp. 1785–1791 Vol. 2.

[5] A. LaTorre, S. Muelas, and J.-M. Pea, "A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test," *Soft Computing*, vol. 15, no. 11, pp. 2187–2199, 2011.

[6] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Computing*, vol. 15, no. 11, pp. 2141–2155, 2011.

[7] P. Menon, J. Kim, D. Bates, and I. Postlethwaite, "Clearance of nonlinear flight control laws using hybrid evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 689–699, 2006.

[8] H. R. Cai, C. Chung, and K. Wong, "Application of differential evolution algorithm for transient stability constrained optimal power flow," *Power Systems, IEEE Transactions on*, vol. 23, no. 2, pp. 719–728, 2008.

[9] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, 2011.

[10] S. Kukkonen and J. Lampinen, "Gde3: the third evolution step of generalized differential evolution," in *The 2005 IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 443–450.

[11] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "Promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization," in *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, ser. GECCO '05. New York, NY, USA: ACM, 2005, pp. 225–232.

[12] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *2007 IEEE Congress on Evolutionary Computation*, 2007, pp. 3523–3530.

[13] R. Gämperle, S. Müller, and P. Koumoutsakos, "A Parameter Study for Differential Evolution," in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, A. Grmela and N. Mastorakis, Eds. WSEAS Press, 2002, pp. 293–298.

[14] K. Zielinski, P. Weitkemper, R. Laur, and K. D. Kammeyer, "Parameter study for differential evolution using a power allocation problem including interference cancellation," in *IEEE Congress on Evolutionary Computation 2006 (CEC 2006)*, 2006, pp. 1857–1864.

[15] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, ser. Natural Computing Series. U.S. Government Printing Office, 2005.

[16] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19 – 31, 2011.

[17] M. Srinivas and L. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656 –667, apr 1994.

[18] T. Bäck, "The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm," in *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*. North-Holland, Amsterdam, 1992.

[19] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *The 2005 conference on Genetic and evolutionary computation (GECCO'05)*. ACM, 2005, pp. 991–998.

[20] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, 2011.

[21] R. Mallipeddi, P. Suganthan, Q. Pan, and M. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679 – 1696, 2011.

[22] J. Zhang and A. Sanderson, "JADE: Adaptive Differential Evolution With Optional External Archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, 2009.

[23] C. Segura, C. A. Coello Coello, E. Segredo, and C. León, "On the adaptation of the mutation scale factor in differential evolution," *Optimization Letters*, no. In press, 2014.

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

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

[26] J. Tvrdík, R. Poláková, J. Veselský, and P. Bujok, "Adaptive variants of differential evolution: Towards control-parameter-free optimizers," in *Handbook of Optimization*, ser. Intelligent Systems Reference Library, I. Zelinka, V. Snášel, and A. Abraham, Eds. Springer Berlin Heidelberg, 2013, vol. 38, pp. 423–449.

[27] K. Zielinski, X. Wang, and R. Laur, "Comparison of adaptive approaches for differential evolution," in *Parallel Problem Solving from Nature PPSN X*, ser. Lecture Notes in Computer Science, G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, Eds. Springer Berlin Heidelberg, 2008, vol. 5199, pp. 641–650.

[28] J. Tvrdík, "Competitive differential evolution," in *MENDEL 2006: 12th International Conference on Soft Computing*, R. Matoušek and P. Ošmera, Eds., University of Technology, Brno, 2006, pp. 7–12.

[29] A. K. Qin, V. L. Huang, and P. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.

[30] F. Peng, K. Tang, G. Chen, and X. Yao, "Multi-start JADE with knowledge transfer for numerical optimization," in *2009 IEEE Congress on Evolutionary Computation*, 2009, pp. 1889–1895.

[31] D. Zaharie, "Control of Population Diversity and Adaptation in Differential Evolution Algorithms," in *Proc. of Mendel 2003, 9th International Conference on Soft Computing*, R. Matousek and P. Osmera, Eds., Brno, Czech Republic, Jun. 2003, pp. 41–46.

[32] A. K. Qin and P. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *The 2005 IEEE Congress onE volutionary Computation*, vol. 2, 2005, pp. 1785–1791 Vol. 2.

[33] M. Lozano, D. Molina, and F. Herrera, "Editorial Scalability of Evolutionary Algorithms and Other Metaheuristics for Large-scale Continuous Optimization Problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, pp. 1–3, 2010.

[34] J. Brest, A. Zamuda, B. Bokovi, S. Greiner, and V. umer, "An analysis of the control parameters adaptation in de," in *Advances in Differential Evolution*, ser. Studies in Computational Intelligence, U. Chakraborty, Ed. Springer Berlin Heidelberg, 2008, vol. 143, pp. 89–110.

[35] W. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 5, pp. 561–578, 2007.

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