An Analysis of the Control Parameters' Adaptation in DE

Janez Brest, Aleš Zamuda, Borko Bošković, Sašo Greiner, and Viljem Žumer

Institute of Computer Science,
Faculty of Electrical Engineering and Computer Science,
University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia
janez.brest@uni-mb.si

Summary. The main goal of this chapter is to present an analysis of how self-adaptive control parameters are being changed during the current evolutionary process. We present a comparison of two distinct self-adaptive control parameters' mechanisms, both using Differential Evolution (DE). The first mechanism has recently been proposed in the jDE algorithm, which uses self-adaptation for F and CR control parameters. In the second one, we integrated the well known self-adaptive mechanism from Evolution Strategies (ES) into the original DE algorithm, also for the F and CR control parameters. Both mechanisms keep the third DE control parameter NP fixed during the optimization process. They both use the same DE strategy, same mutation, crossover, and selection operations, even the same initial population, and they both use self-adaptation at individual level.

1 Introduction

The Differential Evolution (DE) [13, 17, 21] algorithm was proposed by Storn and Price, and since then it has been used during many practical cases. The original DE was modified and many new versions have been proposed [13, 16, 17].

The original DE algorithm keeps all three control parameters fixed during the optimization process. However, there still exists a lack of knowledge on how to obtain reasonably good values for the control parameters of DE, over a given function [16, 22]. The necessity for changing control parameters during the optimization process was confirmed, based on the experiment in [8].

Self-adaptation has proved to be highly beneficial when automatically and dynamically adjusting control parameters. Self-adaptation is usually used in Evolution Strategies [4, 5, 6]. Self-adaptation allows an evolution strategy to adapt itself to any general class of problem, by reconfiguring itself accordingly without any user interaction [2, 3, 12]. DE with self-adaptive control parameters has already been presented in [8, 22].

In this analysis the unconstrained benchmark functions will be used. There are many studies that use DE algorithm in different research areas but, based on our knowledge, there is no current study, regarding the analyses of self-adaptive control parameters in DE algorithm.

U.K. Chakraborty (Ed.): Advances in Differential Evolution, SCI 143, pp. 89–110, 2008.
springerlink.com © Springer-Verlag Berlin Heidelberg 2008

This chapter makes the following contributions: (1) the application of a self-adaptive mechanism from evolution strategies to the original DE algorithm to construct a new version of the self-adaptive DE algorithm; (2) comparative study of the proposed DE algorithm with self-adaptive F and CR control parameters, the jDE algorithm, and the original DE algorithm; (3) analysis of how the control parameters are being changed during the evolutionary process.

The chapter is structured as follows. Section 2 gives an overview of work dealing with DE. Section 3 gives a brief background of the original differential evolution algorithm. Section 4 describes those differential evolution algorithms, which use self-adaptive adjusting control parameters. Two different self-adaptive mechanisms are described. Section 5 presents experimental results on the benchmark functions and gives performance comparisons for the self-adaptive and original DE algorithms. Discussion of the obtained results is given in Section 6. Section 7 concludes the chapter with some final remarks.

2 Work Related to Adaptation in Differential Evolution

Ali and Törn in [1] proposed new versions of the DE algorithm, and also suggested some modifications to the classical DE in order to improve its efficiency and robustness. They introduced an auxiliary population of NP individuals alongside the original population (noted in [1], a notation using sets is used – population set-based methods). Next they proposed a rule for calculating the control parameter F, automatically. Liu and Lampinen [16] proposed a version of DE, where the mutation control parameter and the crossover control parameter are adaptive. Teo in [22] proposed an attempt at self-adapting the population size parameter, in addition to self-adapting crossover and mutation rates. Brest et al. in [8] proposed a DE algorithm, using a self-adapting mechanism on the F and CR control parameters. The performance of the self-adaptive differential evolution algorithm was evaluated on the set of benchmark functions provided for constrained real parameter optimization [10]. In [18] Qin and Suganthan proposed the Self-adaptive Differential Evolution algorithm (SaDE), where the choice of learning strategy and the two control parameters F and CR do not require pre-defining. During evolution, suitable learning strategy and parameter settings are gradually self-adapted, according to the learning experience. Brest et al. [7] reported the performance comparison of certain selected DE algorithms, which use different self-adaptive or adaptive control parameter mechanisms.

In our paper [11] we presented experimental results on how control parameters are being changed during the evolutionary process on the constrained real parameter optimization benchmark functions (CEC2006 [10, 14]).

3 The Original DE Algorithm

In this section we give some background on the DE algorithm [19, 20, 21] that is important for understanding the rest of this chapter.

Differential Evolution (DE) is a floating-point encoding evolutionary algorithm for global optimization over continuous spaces [13, 16, 17, 21], which can also work with discrete variables. DE creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness value. DE has three control parameters: the amplification factor of the difference vector – F, crossover control parameter – CR, and population size – NP.

The general problem an optimization algorithm is concerned with is to find a vector $\mathbf{x}$ so as to optimize $f(\mathbf{x}); \mathbf{x} = (x_1, x_2, ..., x_D)$. D is the dimensionality of the function f. The variables' domains are defined by their lower and upper bounds: $x_{j,low}, x_{j,upp};\ j \in \{1, ..., D\}$. The initial population is selected uniform randomly between the lower ($x_{j,low}$) and upper ($x_{j,upp}$) bounds defined for each variable x_j. These bounds are specified by the user according to the nature of the problem.

DE is a population-based algorithm and vector $\mathbf{x}_{i,G},\ i = 1, 2, ..., NP$ is an individual in the population. NP denotes population size and G the generation. During one generation for each vector, DE employs mutation, crossover and selection operations to produce a trial vector (offspring) and to select one of those vectors with the best fitness value.

By mutation for each population vector a mutant vector $\mathbf{v}_{i,G}$ is created. One of the most popular DE mutation strategy is *'rand/1/bin'* [17, 21]:

$$\mathbf{v}_{i,G} = \mathbf{x}_{r_1,G} + F \times (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}) \tag{1}$$

where the indexes r_1, r_2, r_3 represent the random and mutually different integers generated within the range $[1, NP]$ and also different from index i. F is an amplification factor of the difference vector within the range $[0, 2]$, but usually less than 1.

The original DE algorithm is described very well in literature [17, 21], and, therefore, we will skip a detailed description of the whole DE algorithm.

4 Self-Adaptive DE Algorithms

In this section we describe two different self-adaptive mechanisms of control parameters in the DE algorithm. Both mechanisms use self-adaptation of control parameters at the individual level. The first mechanism uses uniform distribution for changing the values of the control parameter, while the second uses a self-adaptive mechanism found in evolution strategies.

4.1 The Self-Adaptive Control Parameters in a jDE Algorithm

Self-Adaptive DE refers to the self-adaptive mechanism on the control parameters, as proposed by Brest et al. [8]. This self-adapting mechanism uses the already exposed '*rand/1/bin*' strategy (see formula (1)).

In [8] a self-adaptive control mechanism was used to change the control parameters F and CR during the evolutionary process. The third control parameter NP was kept unchanged.

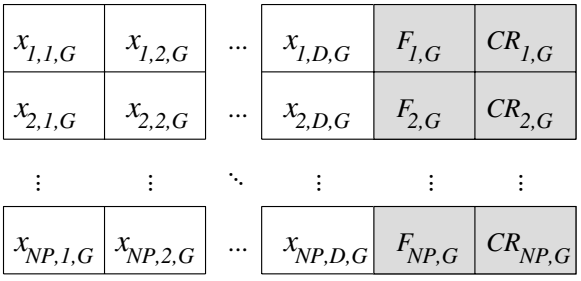

Fig. 1. Self-adapting control parameters F and CR are encoded into the individual. The vector of each individual $\mathbf{x}_{i,G}$ is extended by the values of two control parameters: $F_{i,G}$ and $CR_{i,G}$.

Each individual in the population was extended using the values of these two control parameters (see Figure 1). Both of them were applied at an individual level. Better values for these (encoded) control parameters lead to better individuals which, in turn, are more likely to survive and produce offspring and, hence, propagate these better parameter values.

New control parameters $F_{i,G+1}$ and $CR_{i,G+1}$ were calculated as follows:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \times F_u & \text{if } rand_2 < \tau_1, \\ F_{i,G} & \text{otherwise,} \end{cases} \tag{2}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR_{i,G} & \text{otherwise.} \end{cases} \tag{3}$$

They produce control parameters F and CR in a new vector. $rand_j, j \in \{1, 2, 3, 4\}$ are uniform random values $\in [0, 1]$. τ_1 and τ_2 represent the probabilities of adjusting control parameters F and CR, respectively. τ_1, τ_2, F_l, F_u were taken fixed values $0.1, 0.1, 0.1, 0.9$, respectively. The new F takes a value from $[0.1, 1.0]$ in a random manner. The new CR takes a value from $[0, 1]$. $F_{i,G+1}$ and $CR_{i,G+1}$ are obtained before the mutation is performed, so they influence the mutation, crossover, and selection operations of the new vector $\mathbf{x}_{i,G+1}$.

4.2 The SA-DE Algorithm

As mentioned earlier, evolution strategies [6] are well-known for including a self-adaptive mechanism, encoded directly in each individual of the population. An evolution strategy (ES) has a notation $\mu/\rho, \lambda$-ES, where μ is parent population size, ρ is the number of parents for each new individual, and λ is child population size. An individual is denoted as $\mathbf{a} = (\mathbf{x}, \mathbf{s}, F(\mathbf{x}))$, where $\mathbf{x}$ are search parameters, $\mathbf{s}$ are control parameters, and $F(\mathbf{x})$ is the evaluation of the individual.

We used the idea of self-adaptive mechanism from evolution strategies and applied this idea to the original DE. We shall name the new constructed version of DE, the SA-DE algorithm.

Each individual (see Figure 1) of the SA-DE algorithm is extended to include self-adaptive F and CR control parameters in a similar way as in the jDE algorithm.

A trial vector is composed by mutation and recombination for each individual in population. The mutation procedure is different in the SA-DE algorithm in comparison to the original DE. For adapting the amplification factor of the difference vector F_i for trial individual i, from parent generation G into child generation $G+1$ for the trial vector, the following formula is used:

$$F_{i,G+1} = \langle F_G \rangle_i \times e^{\tau N(0,1)}, \tag{4}$$

where τ denotes the learning factor and is equal to $\tau = 1/\sqrt{2D}$, D being the dimension of the problem. $N(0,1)$ is a random number with a Gauss distribution. The $\langle F_G \rangle_i$ denotes the averaging of the parameters F of individuals i, r_1, r_2, and r_3 from generation G:

$$\langle F_G \rangle_i = \frac{F_{i,G} + F_{r_1,G} + F_{r_2,G} + F_{r_3,G}}{4}. \tag{5}$$

An analogous formula is used for CR of the trial individual i:

$$CR_{i,G+1} = \langle CR_G \rangle_i \times e^{\tau N(0,1)}, \tag{6}$$

where the τ used here is the same as for the adaptation of the F parameter. The $\langle CR_G \rangle_i$ denotes the averaging of the parameters again:

$$\langle CR_G \rangle_i = \frac{CR_{i,G} + CR_{r_1,G} + CR_{r_2,G} + CR_{r_3,G}}{4}. \tag{7}$$

The recombination process is not affected by our strategy, but rather taken from the strategy *'rand/1/bin'* (see Eq. (1)) of the original DE, and the adapted CR_i is used for each individual. The selection principle also helps in adapting F and CR, because only the individuals adapting good parameters can survive.

During the experiments, the following parameter settings were used for the SA-DE algorithm: the global lower and upper bounds for control parameter F were $0.3 \leq F \leq 1.1$, and for control parameter CR were $1/D \leq CR \leq 1$.

5 Experimental Results

The benchmark function test suite used in the experiments for this work, is presented in Table 4. A detailed description about test functions is given in [23]. All the included functions are to be minimised and have the same number of parameters, but they are tested with different number of function evaluations (FES), have different search space domains, and test various optimizer characteristics. Based on these functions, a performance evaluation of the listed algorithms is applied here.

Parameters settings for the jDE and SA-DE algorithms were presented in the previous section. Population size ($NP = 100$) was fixed for all algorithms in all

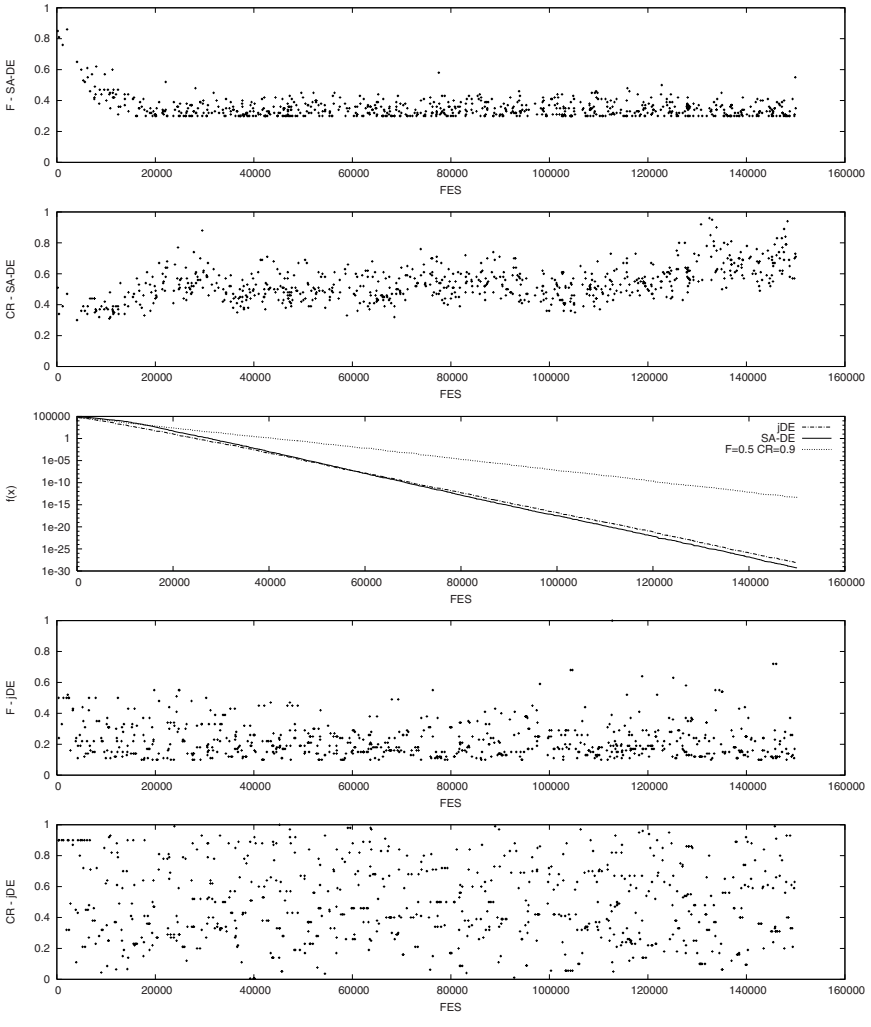

Fig. 2. Values of F and CR control parameters and fitness values of algorithms over one run for function f_1

experiments. As already mentioned, it was of particular interest how the control parameters are being changed during the evolutionary process.

Figures 2–13 show the values for initial parameters F and CR, and the convergence graphs for benchmark functions. Each figure has five sub-figures. Let us describe the sub-figures from the top to the bottom: the first two sub-figures represent the values for the SA-DE algorithm, the first one representing the values of control parameter F and the second the values for the CR control parameter. The fourth and fifth sub-figures represent the same control parameters for

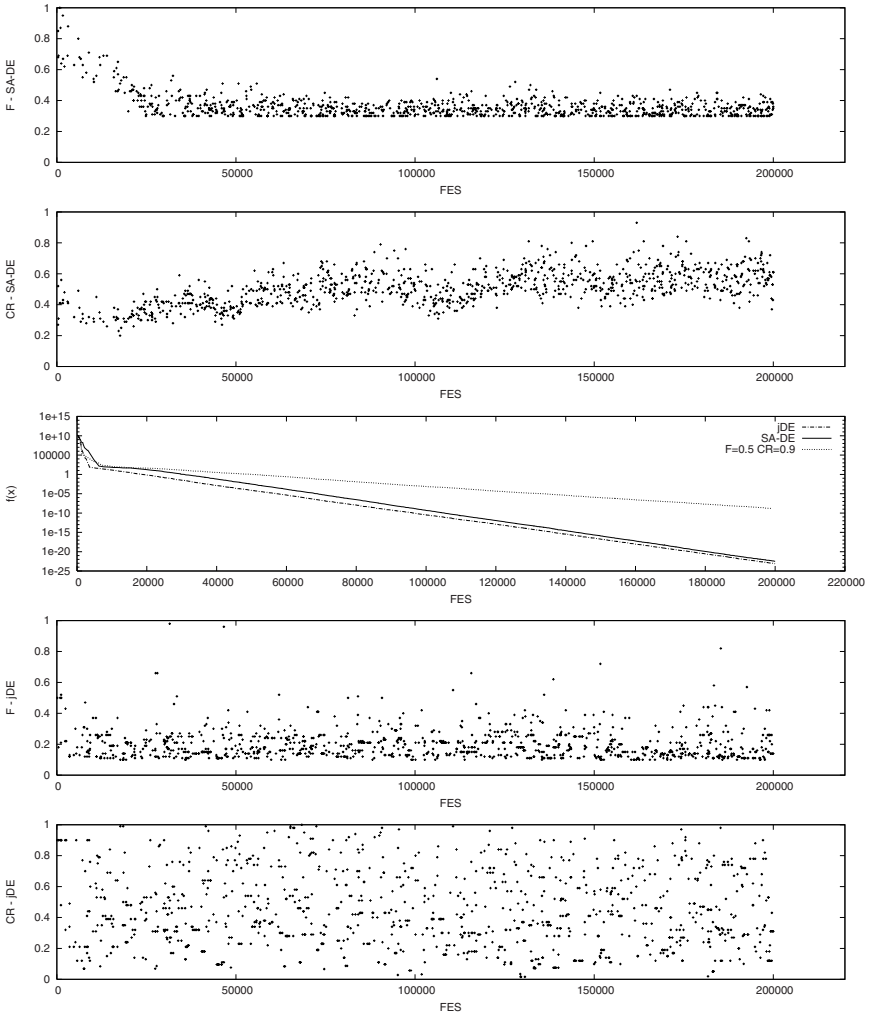

Fig. 3. Values of F and CR control parameters and fitness values of algorithms over one run for function f_2

the jDE algorithm as the first two sub-figures for the SA-DE algorithm. Third sub-figure presents convergence graphs of the fitness values for the jDE, SA-DE, and original DE algorithms. The original DE algorithm used fixed values for control parameters $F = 0.5$ and $CR = 0.9$. All algorithms used the same initial population (same seed for random generator).

Figure 2 shows the results obtained by typical evolutionary run, for function f_1. Both self-adaptive algorithms obtained similar results on the convergence

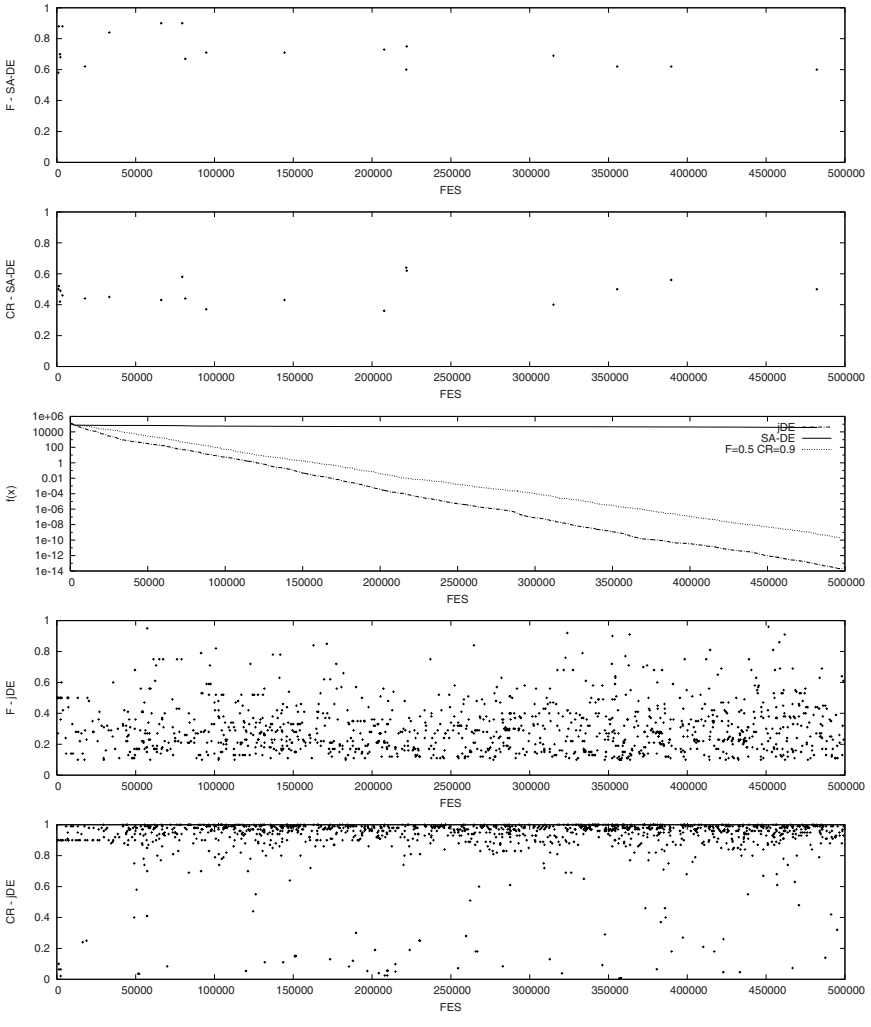

Fig. 4. Values of F and CR control parameters and fitness values of algorithms over one run for function f_3

graph, but the graphs for control parameters F and CR differ. The original DE algorithm obtained the worst results on the fitness convergence graph.

It can be noticed from Figure 3, that convergence graphs for the fitness values regarding the SA-DE and jDE algorithms are very similar (overlapped). The values for control parameter F are, in most cases, less than 0.5 for both algorithms. The original DE algorithm obtained the worst results regarding the fitness convergence graph.

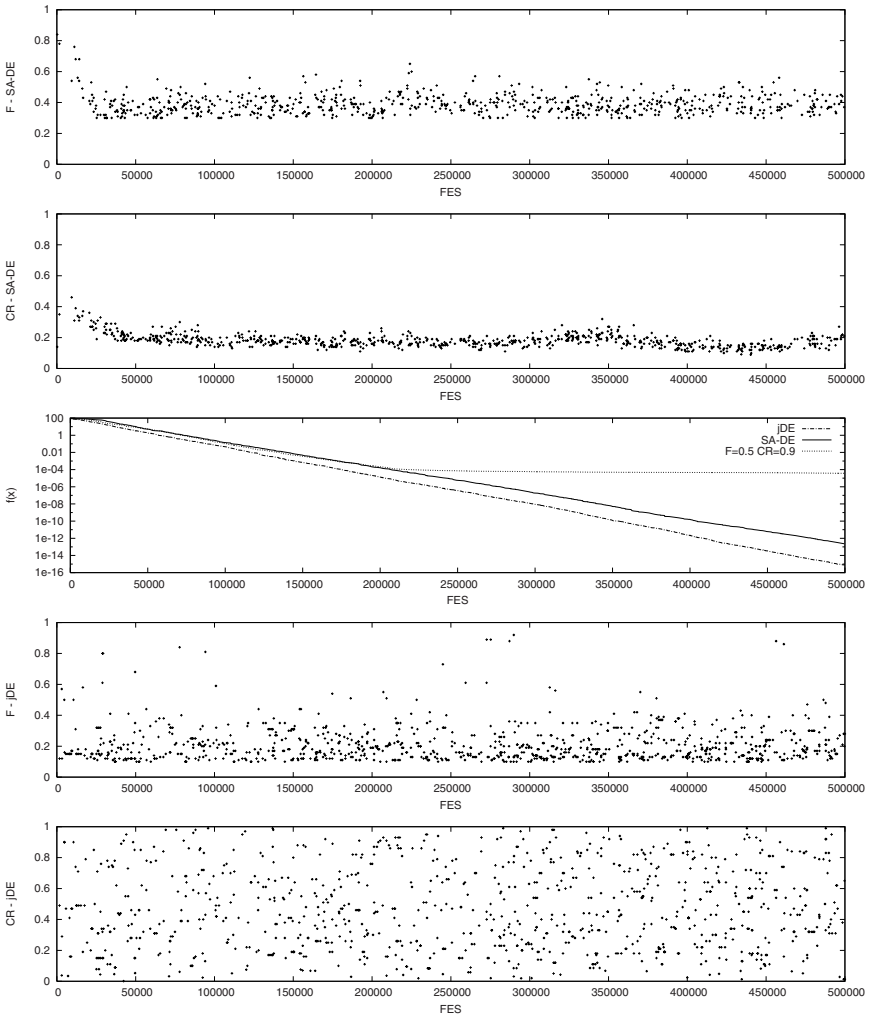

Fig. 5. Values of F and CR control parameters and fitness values of algorithms over one run for function f_4

Figure 4 shows the obtained results for function f_3, where the jDE algorithm outperformed the SA-DE algorithm. A very small number of the currently best fitness value's improvement occurred by the SA-DE algorithm. This algorithm obtained the worst results on the convergence graph.

If we compare the values for the control parameters F and CR of the jDE algorithm in Figures 2–4, for functions f_1, f_2, and f_3, a similarity to the control parameter F can be noticed: there are more values for F less than 0.5. Values for control parameter CR are equally distributed from 0 to 1 for functions f_1

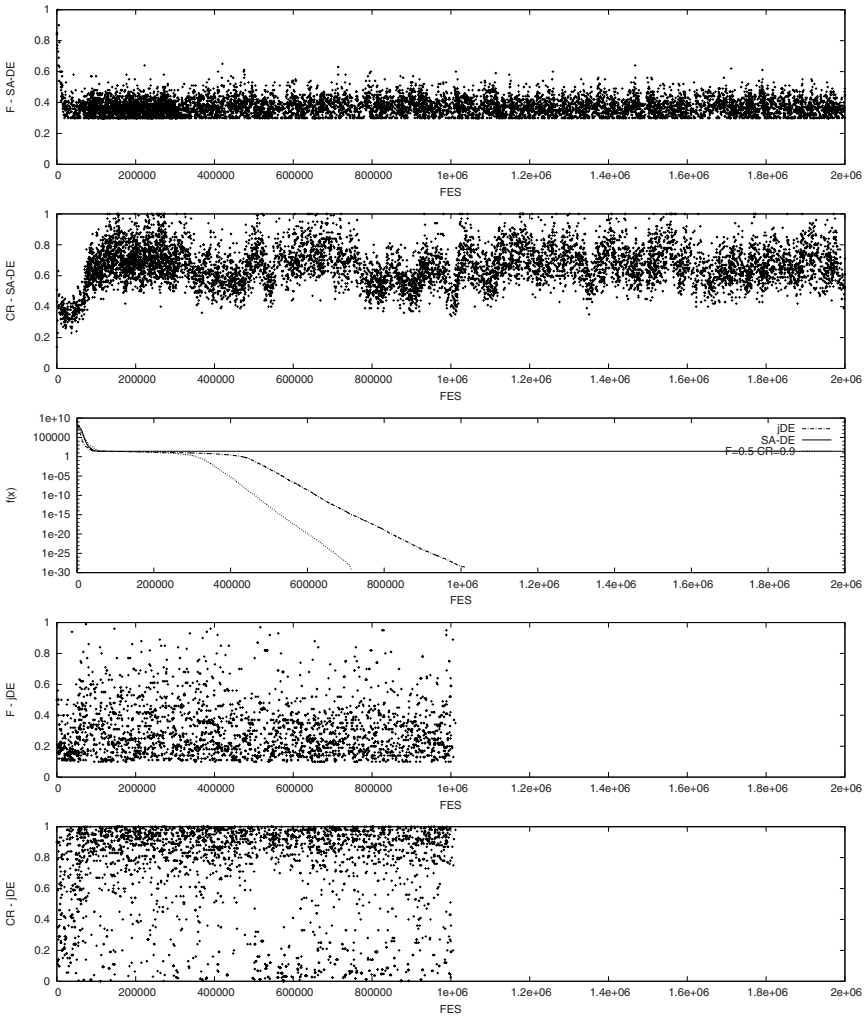

Fig. 6. Values of F and CR control parameters and fitness values of algorithms over one run for function f_5

and f_2, while the values of CR are very high (CR is greater than 0.8) in the case of function f_3.

The jDE and SA-DE algorithms obtained quite similar results on convergence graphs for function f_4 (see Figure 5). The jDE algorithm performed slightly better. The original DE algorithm obtained the worst results on the fitness convergence graph. It did not obtain much improvement in the fitness values after a half of the predefined maximum number of function evaluations was reached.

Figure 6 shows the obtained results for function f_5. In this case the best results ware obtained by the original DE algorithm, followed by the jDE algorithm.

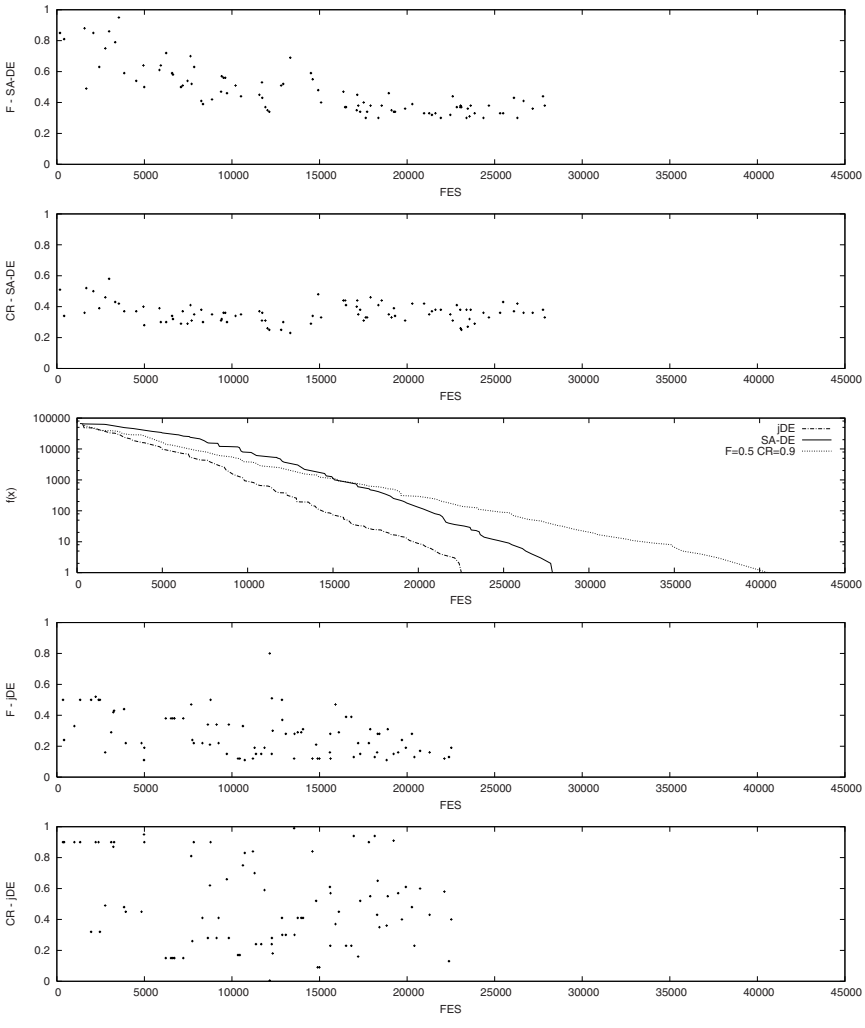

Fig. 7. Values of F and CR control parameters and fitness values of algorithms over one run for function f_6

The SA-DE algorithm got trapped in local optimum and, therefore obtained the worst result on the convergence graph. It can be noticed that both SA-DE and jDE algorithms conducted a great number of improvements of the currently best individual fitness value during the evolutionary process. CR values are usually high ($CR > 0.7$) for the jDE algorithm.

Figure 7 shows that all algorithms succeeded in solving function f_6. A slightly better performance was obtained by the jDE algorithm, followed by the SA-DE algorithm.

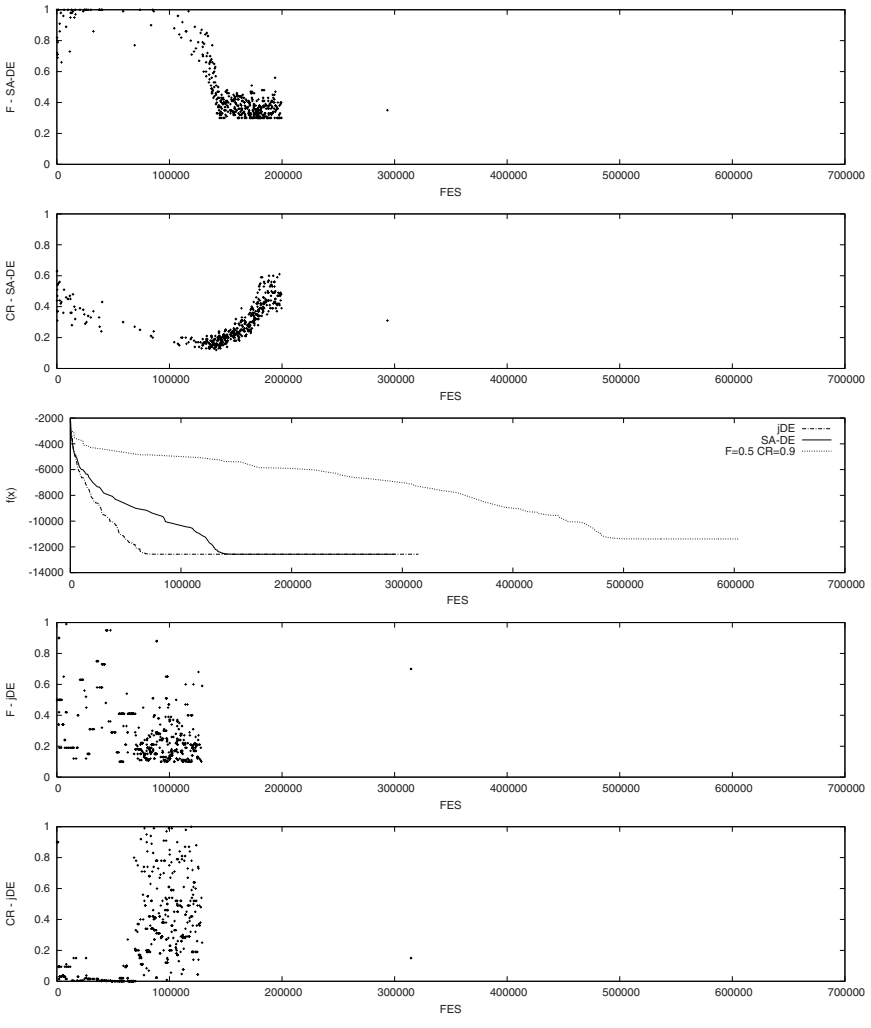

Fig. 8. Values of F and CR control parameters and fitness values of algorithms over one run for function f_8

The self-adaptive algorithms jDE and SA-DE performed much better than the original DE algorithm regarding function f_8 (see Figure 8). The original DE algorithm did not even get close to global optimum. It found the fitness value $-$ 11382.06.

Figure 9 shows the results for function f_9. The best performance results for function f_9 were obtained by the jDE algorithm. The worst performance was obtained by the original DE algorithm. When comparing sub-figures with F and CR for self-adaptive algorithms, it can seen that CR values by both algorithms are very low ($CR < 0.2$).

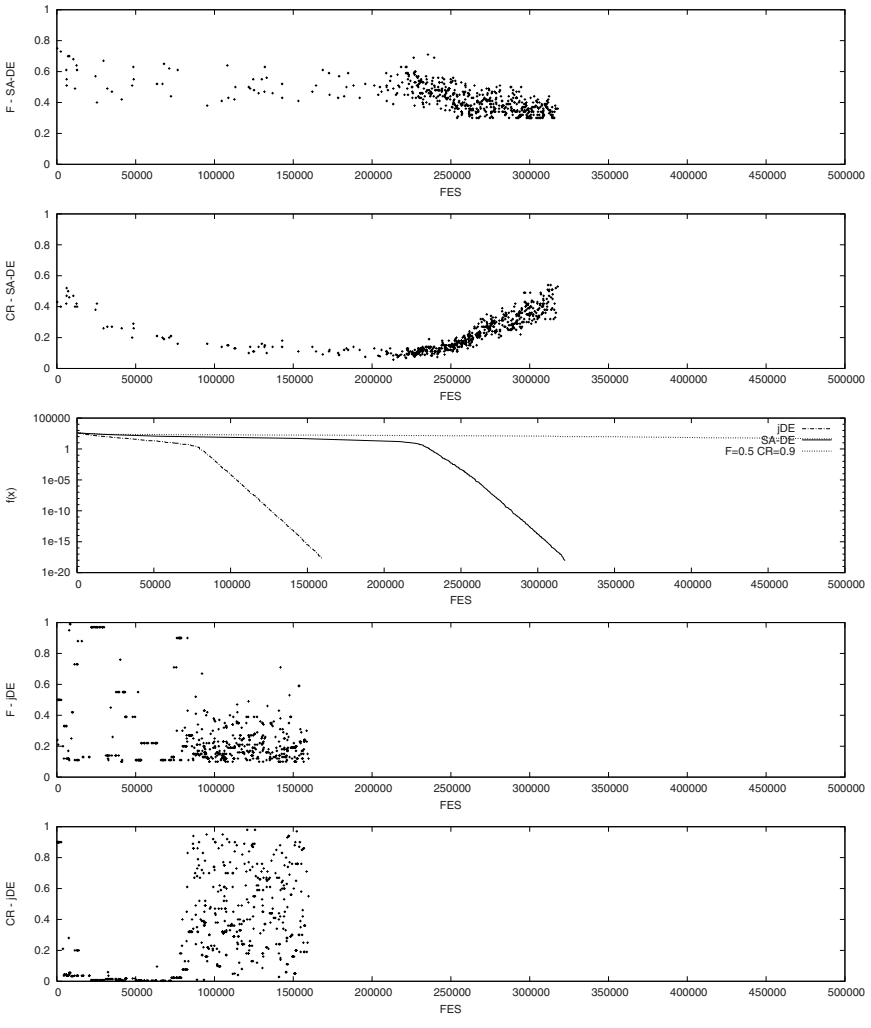

Fig. 9. Values of F and CR control parameters and fitness values of algorithms over one run for function f_9

Figure 9 shows that the jDE algorithm performed best for function f_{10}. If we look at Figures 8–10, it can be noticed that more improvements to the currently best individual occurred by the SA-DE algorithm after approximately 100 000, 200 000 and 70 000 FES for functions f_8, f_9, and f_{10}, respectively.

Figures 11–13 show the results for functions f_{11}, f_{12}, and f_{13}, respectively. For those functions both self-adaptive algorithms obtained better performance than the original DE algorithm. The values for the F control parameter were less than 0.5, while the CR values were equally distributed between 0 and 1 for

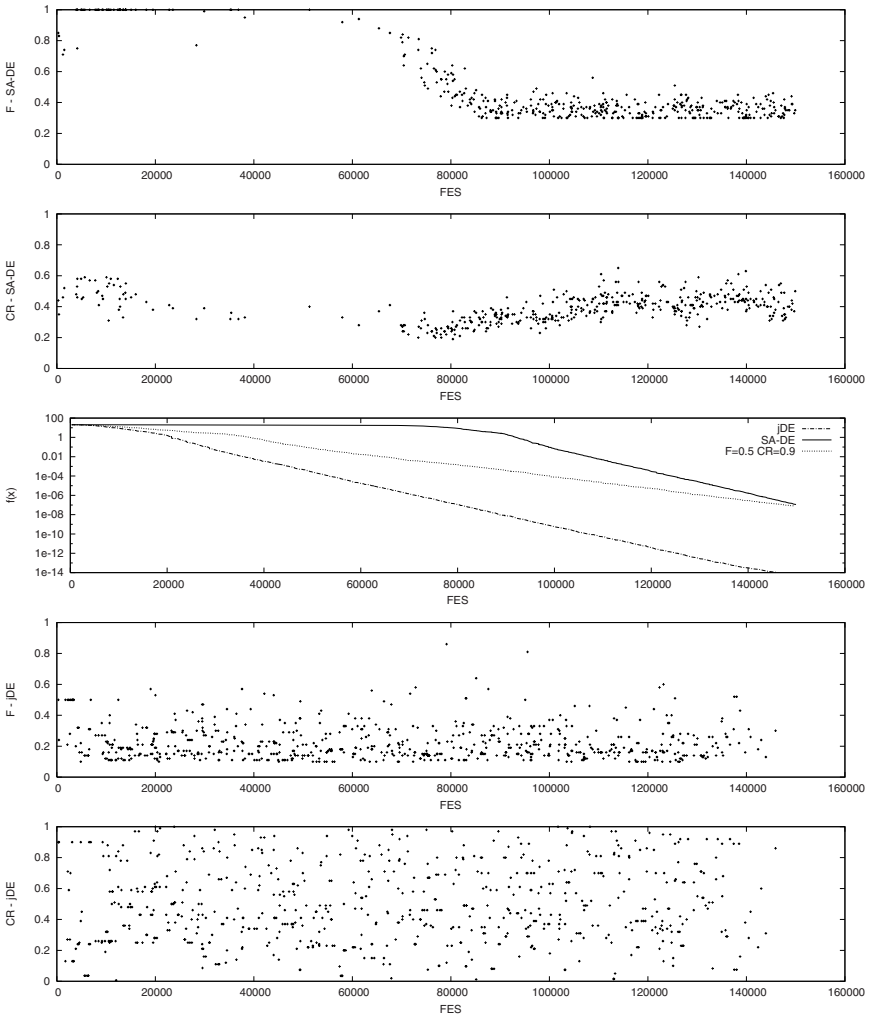

Fig. 10. Values of F and CR control parameters and fitness values of algorithms over one run for function f_{10}

functions f_{11}–f_{13}. For these functions the F values were usually less than 0.5 for the SA-DE algorithm, while the CR values were between 0.3 and 0.7.

The most important conclusion based on the results from Figures 2–13 is that the F and CR values obtained by the self-adaptive jDE and SA-DE algorithms differ. Actually, they also differ for functions, where the convergence graph shows almost equal algorithm performances (f_1, f_2, f_4, f_{11}, f_{12}, and f_{13}).

Figures 2–13 show the obtained results for one typical run of algorithms. Our description of the obtained results is only one point of view on how the control parameters of self-adaptive DE algorithms are being changed during the

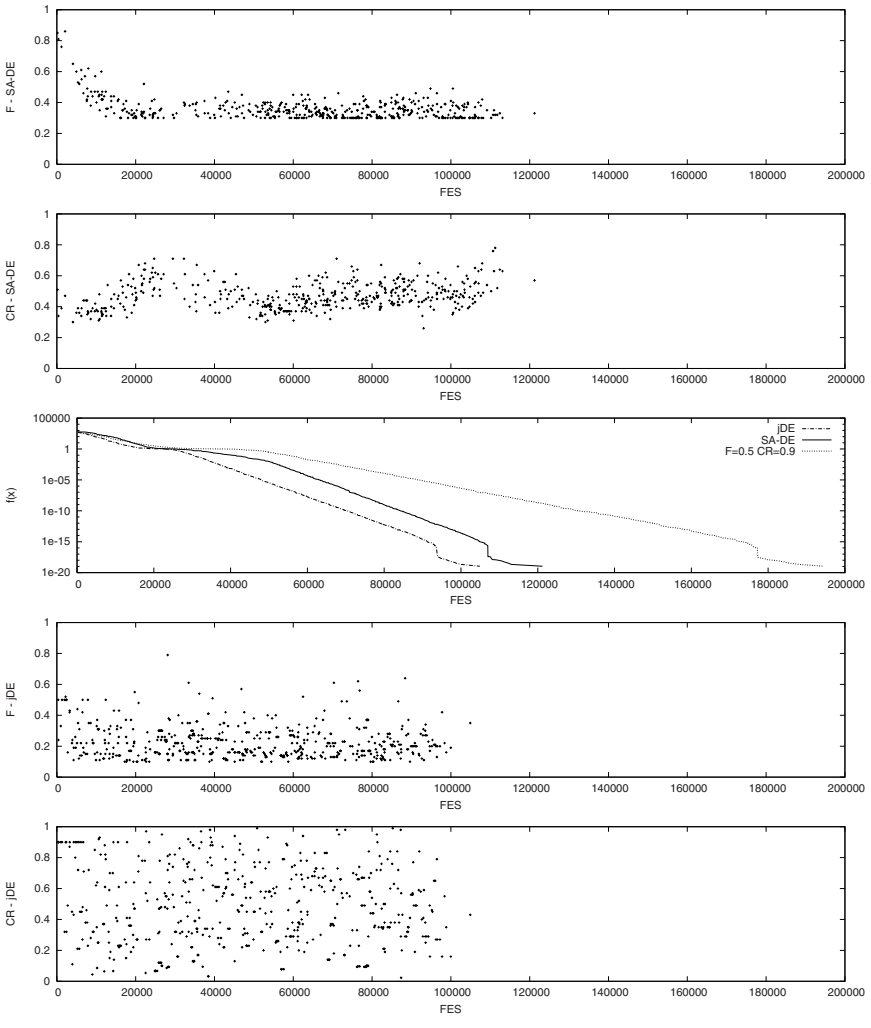

Fig. 11. Values of F and CR control parameters and fitness values of algorithms over one run for function f_{11}

optimization process, and on how changes in control parameters F and CR have an influence on the fitness value of the convergence graph.

Table 1 shows the obtained results for the three algorithms. The jDE algorithm performed well, on average. It obtained the best results for some benchmark functions, but it did not optimize the function f_5 so well, because it got trapped in a local optimum once. Similar observations were gathered for function f_{12}. The SA-DE algorithm gets the best results for the functions f_1, f_{12}, and f_{13}, while it has the worst results for functions f_3, f_8 (only a small number of missed global

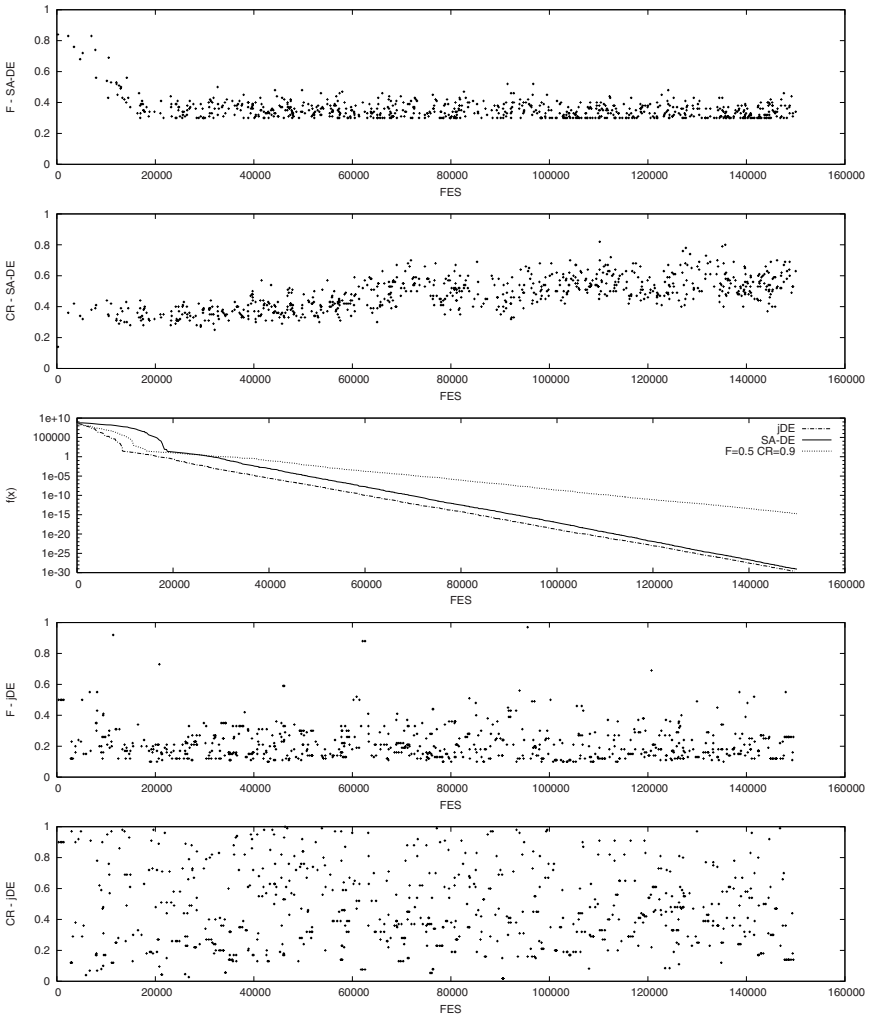

Fig. 12. Values of F and CR control parameters and fitness values of algorithms over one run for function f_{12}

optima), and f_5. The original DE algorithm performs well, but convergence speed is lower than for the self-adaptive algorithms on some benchmark functions.

Table 2 shows the average values for the control parameters F and CR, obtained in the experiment during one evolutionary run. The average values for F using the jDE algorithm are between 0.2 and 0.35, while CR values are more evenly distributed over the $[0, 1]$ interval. For the functions f_8 and f_9, the values are both around 0.35. For the function f_3, the value of the CR control parameter is approximately 0.9, and for the other functions, the CR values are around

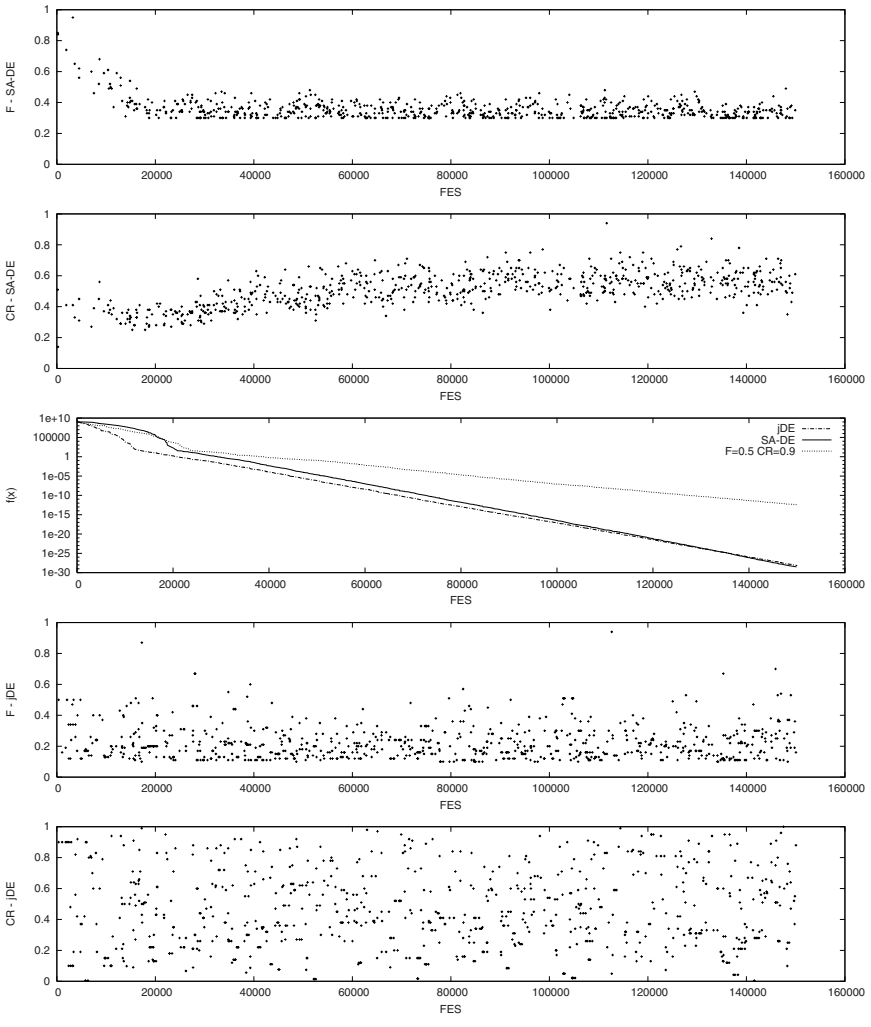

Fig. 13. Values of F and CR control parameters and fitness values of algorithms over one run for function f_{13}

0.5. Remember that the initial values for the control parameters using the jDE algorithms were $F = 0.5$ and $CR = 0.9$.

The next algorithm in Table 2 is the SA-DE algorithm. After evolutionary process, the obtained average F and CR were quite different for each function. For functions f_1, f_2, f_4, f_5, f_{11}, f_{12}, and f_{13}, the obtained average F was between $[0.35, 0.39]$. On these functions, SA-DE was quite successful, with the exception of f_5. For these functions, an average CR parameter was also quite similar, between $[0.47, 0, 54]$, with f_5 again being an exception. A greater CR indicates that many parameters still have to be changed to reach global optimum. In

Table 1. The experimental results, averaged over 100 independent runs, of the jDE algorithm, SA-DE algorithm, and the original DE algorithm ($F = 0.5$, $CR = 0.9$ 'Mean' indicates the average of the minimum best values obtained and 'Std.Dev' stands for the standard deviation

Fun.	Gen.	jDE Mean (Std.Dev)	SA-DE Mean (Std.Dev)	DE $F_{0.5}CR_{0.9}$ Mean (Std.Dev)
f_1	1500	2.83e-28 (2.54e-28)	2.61e-29 (2.97e-29)	8.79e-14 (5.83e-14)
f_2	2000	1.51e-23 (9.13e-24)	4.08e-23 (4.02e-23)	1.42e-9 (9.95e-9)
f_3	5000	6.47e-14 (1.25e-13)	21645 (16103)	6.25e-11 (6.64e-11)
f_4	5000	2.08e-15 (3.18e-15)	5.32e-13 (6.31e-13)	7.35e-2 (1.17e-1)
f_5	20000	0.039 (0.02)	26.46 (7.24)	4.21e-31 (2.27e-30)
f_6	1500	0 (0)	0 (0)	0 (0)
f_7	3000	0.0031 (0.0009)	0.0038 (0.00086)	0.0046 (0.0014)
f_8	9000	−12569.5 (1.07e-11)	−12568.3 (11.84)	−11148.5 (496.6)
f_9	5000	0 (0)	0 (0)	68.18 (33.67)
f_{10}	1500	8.73e-15 (2.54e-15)	1.18e-05 (8.09e-05)	9.97e-8 (4.13e-8)
f_{11}	2000	0 (0)	0 (0)	7.39e-5 (7.39e-4)
f_{12}	1500	6.74e-30 (8.15e-30)	2.84e-29 (4.12e-29)	7.82e-15 (7.79e-15)
f_{13}	2000	1.24e-28 (1.44e-28)	1.02e-28 (1.33e-28)	5.31e-14 (5.76e-14)

Table 2. Average F and CR in a typical run of each algorithm

Fun.	jDE F	jDE CR	SA-DE F	SA-DE CR
f_1	0.22±0.11	0.50±0.25	0.35±0.06	0.54±0.11
f_2	0.21±0.12	0.45±0.24	0.36±0.08	0.51±0.11
f_3	0.31±0.18	0.90±0.19	0.72±0.11	0.48±0.08
f_4	0.22±0.13	0.44±0.25	0.39±0.07	0.18±0.04
f_5	0.30±0.17	0.69±0.32	0.38±0.06	0.67±0.13
f_6	0.21±0.11	0.49±0.27	0.47±0.16	0.36±0.06
f_7	0.24±0.10	0.54±0.28	0.48±0.13	0.36±0.05
f_8	0.32±0.23	0.36±0.30	0.48±0.21	0.30±0.12
f_9	0.24±0.15	0.35±0.29	0.43±0.09	0.23±0.12
f_{10}	0.23±0.12	0.48±0.25	0.45±0.21	0.39±0.09
f_{11}	0.23±0.11	0.47±0.25	0.36±0.07	0.47±0.09
f_{12}	0.22±0.11	0.48±0.25	0.36±0.06	0.48±0.11
f_{13}	0.22±0.11	0.47±0.25	0.36±0.07	0.51±0.11

the case of f_4, a CR drop allows a much more precise selection, which makes the algorithm perform better than the original DE on this function, because the overall function evaluation improvement is achieved by diminishing each x_i component. Another pattern can be observed with the functions f_6–f_{10}. The average F is between $[0.43, 0.48]$ here, and CR is smaller and between $[0.23, 0.39]$. For all these functions, the performance of the SA-DE algorithm is the same or better than the original DE, except for the f_8, where the convergence is not as rapid as with the other two algorithms.

Table 3. Number of improvements

Fun.	jDE	SA-DE	DE $F_{0.5}$ $CR_{0.9}$
f_1	805	726	353
f_2	1024	985	411
f_3	1224	19	387
f_4	898	729	870
f_5	2833	8106	1561
f_6	106	93	80
f_7	53	56	39
f_8	415	402	364
f_9	516	529	43
f_{10}	680	440	325
f_{11}	525	449	406
f_{12}	763	734	287
f_{13}	690	709	308

For the function f_3, which is the worst case for SA-DE, F and CR still remained quite high, keeping the algorithm from converging into a local optimum, thus promising a potential global optimum convergence.

Table 3 shows how many times the currently best individual was changed. The results were obtained during the same experiment used to obtain the results for control parameters F and CR, as reported in Table 2. From Table 3 it can be noticed that the original DE usually improved the currently best individual fewer times than for other algorithms. If we compare only both self-adaptive algorithms, the number of improvements is quite similar except for functions f_3 and f_5, where the SA-DE algorithm performed either little or high numbers of improvements, respectively. In both cases the jDE algorithm obtained better results (see Table 1).

The best setting for control parameters is problem dependent. Self-adaptation may help an algorithm to have higher convergence speed to global optimum. The results in this section show that no algorithm performed superiorly better than any other algorithm for all optimization problems.

6 Discussion

When introducing the SA-DE algorithm, we did not make fine-tunings of the τ learning parameter, which is still open for further research. The τ could have been separately defined for F and CR, or it could even be self-adapted, projecting several new experimental combinations to try out. Another constraint that could be changed or alleviated is our initialization phase and the bounds, mostly for F, in SA-DE, where the bounds could be extended or dynamically adapted. As has been confirmed in [15], the lower bound of F has indeed a strong impact on the algorithm convergence.

The SA-DE control parameters crossover process could also be changed to include all the members (in ES, called global intermediate – GI) in the population or the few random best ones.

Other possibilities for self-adaptation would be at the component level, where each component x_i of each individual would have its own control parameters. Another possibility is to encode control parameters at the population level, where same parameters are used for one generation – similarly, but not the same as the GI approach. We have indeed tried many combinations of these proposals, confirming that many research opportunities are still open.

The presented control parameters analysis did not include the NP parameter adaptation, which is also a candidate for future research. Readers are referred to our recent work [9].

7 Conclusion

The chapter presents two self-adaptive mechanisms in the DE. Both mechanisms are implemented at the individual level. Self-adaptation may help an algorithm to perform better for convergence speed, and an algorithm with self-adaptation may have greater robustness, on average.

Our goal in this work was not to make fine tunning of each self-adaptive mechanism to obtain the best result for a particular optimization problem, but rather to give some ideas on how to apply self-adaptive control parameters (F and CR) in a DE algorithm, in order to achieve better performance, in general.

References

[1] Ali, M.M., Törn, A.: Population Set-Based Global Optimization Algorithms: Some Modifications and Numerical Studies. Computers & Operations Research 31(10), 1703–1725 (2004)
[2] Bäck, T.: Adaptive Business Intelligence Based on Evolution Strategies: Some Application Examples of Self-Adaptive Software. Information Sciences 148, 113–121 (2002)
[3] Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Institute of Physics Publishing and Oxford University Press (1997)
[4] Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, New York (1996)
[5] Beyer, H.-G.: The theory of Evolution Strategies. Springer, Berlin (2001)
[6] Beyer, H.-G., Schwefel, H.-P.: Evolution strategies: a comprehensive introduction. Natural Computing 1, 3–52 (2002)
[7] Brest, J., Bošković, B., Greiner, S., Žumer, V., Sepesy Maučec, M.: Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Computing - A Fusion of Foundations, Methodologies and Applications 11(7), 617–629 (2007)

[8] Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, 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)
[9] Brest, J., Sepesy Maučec, M.: Population Size Reduction for the Differential Evolution Algorithm. Applied Intelligence (accepted) DOI: 10.1007/s10489-007-0091-x
[10] Brest, J., Žumer, V., Sepesy Maučec, M.: Self-adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In: The 2006 IEEE Congress on Evolutionary Computation CEC 2006, pp. 919–926. IEEE Press, Los Alamitos (2006)
[11] Brest, J., Žumer, V., Maučec, M.S.: Control Parameters in Self-Adaptive Differential Evolution. In: Filipič, B., Šilc, J. (eds.) Bioinspired Optimization Methods and Their Applications, Ljubljana, Slovenia, October 2006, pp. 35–44. Jožef Stefan Institute (2006)
[12] Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. In: Natural Computing. Springer, Berlin (2003)
[13] Feoktistov, V.: Differential Evolution: In Search of Solutions. Springer, New York (2006)
[14] Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, N., Coello, C.A.C., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical Report Report #2006005, Nanyang Technological University, Singapore and et al. (December 2005), `http://www.ntu.edu.sg/home/EPNSugan`
[15] Liang, K.-H., Yao, X., Newton, C.S.: Adapting self-adaptive parameters in evolutionary algorithms. Appl. Intell. 15(3), 171–180 (2001)
[16] Liu, J., Lampinen, J.: A Fuzzy Adaptive Differential Evolution Algorithm. Soft Computing - A Fusion of Foundations, Methodologies and Applications 9(6), 448–462 (2005)
[17] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution, A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
[18] Qin, A.K., Suganthan, P.N.: Self-adaptive Differential Evolution Algorithm for Numerical Optimization. In: The 2005 IEEE Congress on Evolutionary Computation CEC 2005, vol. 2, pp. 1785–1791. IEEE Press, Los Alamitos (2005)
[19] Rönkkönen, J., Kukkonen, S., Price, K.V.: Real-Parameter Optimization with Differential Evolution. In: The 2005 IEEE Congress on Evolutionary Computation CEC 2005, vol. 1, pp. 506–513. IEEE Press, Los Alamitos (2005)
[20] Storn, R., Price, K.: Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA (1995)
[21] Storn, R., Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341–359 (1997)
[22] Teo, J.: Exploring dynamic self-adaptive populations in differential evolution. Soft Computing - A Fusion of Foundations, Methodologies and Applications 10(8), 673–686 (2006)
[23] Yao, X., Liu, Y., Lin, G.: Evolutionary Programming Made Faster. IEEE Transactions on Evolutionary Computation 3(2), 82–102 (1999)

Appendix

Table 4. Benchmark functions used in this study

Test function	D	S	f_{min}
$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]^D$	0
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	30	$[-10, 10]^D$	0
$f_3(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]^D$	0
$f_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le D\}$	30	$[-100, 100]^D$	0
$f_5(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]^D$	0
$f_6(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-100, 100]^D$	0
$f_7(x) = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]^D$	0
$f_8(x) = \sum_{i=1}^{D} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]^D$	-12569.5
$f_9(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]^D$	0
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos 2\pi x_i\right) + 20 + e$	30	$[-32, 32]^D$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^D$	0
$f_{12}(x) = \frac{\pi}{D}\{10\sin^2(\pi y_i) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i - a)^m, & x_i < -a. \end{cases}$	30	$[-50, 50]^D$	0
$f_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_D - 1)^2[1 + \sin^2(2\pi x_D)]\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	30	$[-50, 50]^D$	0