A Simple Adaptive Differential Evolution Algorithm

Radha Thangaraj[1], Millie Pant[1] and Ajith Abraham[2]
[1]Indian Institute of Technology Roorkee, India
[2]Norwegian University of Science and Technology, Norway
t.radha@ieee.org, millifpt@iitr.ernet.in, ajith.abraham@ieee.org

Abstract— **Differential Evolution (DE) is a simple and efficient scheme for global optimization over continuous spaces. DE is generally considered as a reliable, accurate, robust and fast optimization techniques. It outperforms many other optimization algorithms in terms of convergence speed and robustness over common benchmark problems and real world applications. However, the user is required to set the values of the control parameters of DE for each problem. Such parameter tuning is a time consuming task. In this paper, a new Differential Evolution algorithm based on Adaptive Control parameters (ACDE) is introduced. The performance of ACDE algorithm is investigated with ten standard benchmark problems and the results are compared with the classical DE algorithm in terms of average fitness function value, number of function evaluations, convergence time and success rate. The numerical results show that the ACDE algorithm outperforms the classical DE in terms of all considered performance measures.**

Keywords-Differential Evolution; global optimization; control parameters;

I. INTRODUCTION

Differential Evolution is a stochastic, population based search strategy developed by Price and Storn [1]. It is a novel evolutionary approach capable of handling non-differentiable, non-linear and multi-modal objective functions. DE has been consistently ranked as one of the best search algorithm for solving global optimization problems in several case studies. DE has been designed as a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of EAs. The method typically requires few, easily chosen control parameters. Experimental results have shown that performance of DE is better than many other well known EAs [2], [3]. While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process [4]. Mutation operation plays the most significant role in the performance of a DE algorithm.

Despite having several attractive features, it has been observed that DE sometimes does not perform as good as the expectations. Empirical analysis of DE has shown that it may stop proceeding towards a global optimum even though the population has not converged even to a local optimum [5]. The situation when the algorithm does not show any improvement though it accepts new individuals in the population is known as stagnation. Besides this, DE also suffers from the problem of premature convergence. This situation arises when there is a loss of diversity in the population. It generally arises when the objective function is multi objective having several local and global optimums. Like other EA, the performance of DE deteriorates with the increase in dimensionality of the objective function. Several modifications have been made in the structure of DE to improve its performance. One class of modification deals with the development of adaptive control parameters [9] – [15]. Use of self adaptive parameters saves the user from the trouble of fine tuning of parameters, which is a crucial task in Evolutionary Algorithms including DE. In this paper we introduce a new Differential Evolution algorithm with Adaptive Control parameters namely ACDE.

The rest of the paper is organized as follows: Section II gives a brief survey on DE family of adaptive control parameter algorithms. Section III and IV discusses the basic DE and the proposed ACDE algorithms respectively. In section V, results and discussion are given; finally the paper concludes with section VI.

II. RELATED WORKS

There are quite different conclusions about the rules for choosing the ***control parameters*** of DE. Price and Storn [1] stated that the control parameters of DE are not difficult to choose. On the other hand, Gämperle et al. [6] reported that choosing the proper control parameters for DE is more difficult than expected. Liu and Lampinen [7] reported that effectiveness, efficiency, and robustness of the DE algorithm are sensitive to the settings of the control parameters. The best settings for the control parameters can be different for different functions and the same function with different requirements for consumption time and accuracy. However, there still exists a lack of knowledge on how to find reasonably good values for the control parameters of DE for a given function [8].

Das et al. [9] introduced two schemes for adapting the scale factor F in DE. In the first scheme they varied F randomly between 0.5 and 1.0 in successive iterations. They suggested decreasing F linearly from 1.0 to 0.5 in their second scheme. This encourages the individuals to sample diverse zones of the search space during the early stages of the search. During the later stages, a decaying scale factor helps to adjust the movements of trial solutions finely so that they can explore the interior of a relatively small space in which the suspected global optimum lies. Teo [10] proposed an attempt at self-adapting the population size parameter in addition to self-adapting crossover and mutation rates. Brest et al. [11], [12] encoded control parameters F and Cr into the individual and evolved their values by using two new

probabilities $\tau_1$and τ_2. In their algorithm (called SADE), a set of F values was assigned to each individual in the population. With probability τ_1, F is reinitialized to a new random value in the range of [0.1, 1.0], otherwise it is kept unchanged. The control parameter Cr, assigned to each individual, is adapted in an identical fashion, but with a different re-initialization range of [0, 1] and with the probability τ_2. With probability τ_2, Cr takes a random value in [0, 1], otherwise it retains its earlier value in the next generation. Differential Evolution with Preferential Crossover (DEPC) was suggested by M. M. Ali in 2007 [13]. In his work he suggested three changes in the basic DE structure. The DEPC algorithm uses F_i as a random variable in $[-1,-0.4] \cup [0.4,1]$ for each targeted point. Secondly DEPC used two population sets S_1 and S_2 containing N points. The function of the auxiliary set S_2 in DEPC is to keep record of the trial points that are discarded in DE. Potential trial points in S_2 are then used for further explorations. Finally DEPC differs used a new crossover rule, namely the preferential crossover, that always generates feasible trial points. Ali tested his algorithm on comprehensive set of benchmark problems and showed that DEPC outperforms the basic DE in most of the test cases.

Yang et al. [14] proposed a self adaptive differential evolution algorithm with neighborhood search (SaNSDE). SaNSDE proposes three self-adaptive strategies: self adaptive choice of the mutation strategy between two alternatives, self-adaptation of the scale factor *F*, and self-adaptation of the crossover rate *Cr*. Qin et al [15] proposed a Self-adaptive DE algorithm (SaDE), where the choice of learning strategy and the two control parameters F and CR are not required to be pre-defined. During evolution, the suitable learning strategy and parameter settings are gradually self-adapted according to the learning experience. Many of the developments in DE algorithm design and applications can be found in [16].

III. Differential Evolution

A general DE variant may be denoted as DE/X/Y/Z, where X denotes the vector to be mutated, Y specifies the number of difference vectors used and Z specifies the crossover scheme which may be binomial or exponential. Throughout the study we shall consider the mutation strategy DE/rand/1/bin [1]. It is also known as the classical version of DE and is perhaps the most frequently used version of DE. DE works as follows: First, all individuals are initialized with uniformly distributed random numbers and evaluated using the fitness function provided. Then the following are executed until maximum number of generation has been reached or an optimum solution is found.

In a D-dimensional search space, for each target vector $x_{i,g}$, a mutant vector is generated by

$$v_{i,g+1} = x_{r_1,g} + F*(x_{r_2,g} - x_{r_3,g}) \qquad (1)$$

where $r_1, r_2, r_3 \in \{1,2,....,NP\}$ are randomly chosen integers, which are different from each other and also different from the running index i. F (>0) is a scaling factor which controls the amplification of the differential vector $(x_{r_2,g} - x_{r_3,g})$.

Once mutation phase is complete, crossover is introduced in order to increase the diversity of the perturbed parameter vectors. The parent vector is mixed with the mutated vector to produce a trial vector $u_{ji,g+1}$,

$$u_{j,i,g+1} = \begin{cases} v_{j,i,g+1} & if \quad rand_j \le Cr \vee j = k \\ x_{j,i,g} & otherwise \end{cases} \qquad (2)$$

where j , k $\in$ {1, 2,……, D}; k is a random parameter index, chosen once for each i . $rand_j \in [0,1]$; Cr is the crossover constant takes values in the range [0, 1].

Finally selection takes place where a tournament is held between the target vector and trial vector and the one with better fitness function is allowed to enter the next generation. In this way individuals in a new generation are as good as or better than the individuals in the previous generation.

IV. Proposed ACDE Algorithm

In this Section, we describe the adaptive Differential Evolution (ACDE) algorithm. The only structural difference between the proposed ACDE algorithm and the basic DE is selecting the control parameters only. DE has only three control parameters; they are the scale factor F, the crossover rate Cr and the population size. Choosing suitable control parameter values is, frequently, a problem-dependent task. The trial-and-error method used for tuning the control parameters requires multiple optimization runs. Moreover in the modified versions of DE (other than adaptive ones), the control parameters are kept fixed throughout the algorithm. Varying the values of control parameters in successive generations will provide more randomness to the algorithm which in turn may help in improving the working of algorithm in terms of exploration and exploitation.

Although we need to fix the values for some parameters in the proposed ACDE algorithm, it can be called adaptive in the sense that in every generation the values of control parameters F and Cr change according to some simple rules which are defined as follows:

$$F_{g+1} = \begin{cases} F_l + rand_1\sqrt{Grand_1^{\,2} + Grand_2^{\,2}} & if \quad P_F < rand_2 \\ F_0 & otherwise \end{cases} \qquad (3)$$

$$Cr_{g+1} = \begin{cases} Cr_l * rand_3 & if \quad P_{Cr} < rand_4 \\ Cr_0 & otherwise \end{cases} \qquad (4)$$

With the help of equations (3) and (4) we calculate the parameters F and Cr for a new generation.

Here, $rand_j$, $j \in \{1,2,3,4\}$ are uniform random numbers in the interval (0, 1]. $Grand_1$ and $Grand_2$ are Gaussian distributed random numbers with mean 0 and standard deviation 1. P_F and P_{Cr} are the probabilities to adjust the factors F and Cr respectively. In the present study, we have taken $P_F = P_{Cr} = 0.5$. Values of the constants F_l, F_0, Cr_l, Cr_0 are taken as $F_l = Cr_l = 0.1$, $F_0 = Cr_0 = 0.5$. The scaling factor

F is kept within the range F_u and F_l where $F_u = 0.5$ and F_l is 0. We have used the following bounds for F.

$$\begin{aligned} &If \quad F_{g+1} > F_u \quad then \quad F_{g+1} = F_u * rand_5 \\ &If \quad F_{g+1} < F_l \quad then \quad F_{g+1} = F_l * rand_6 \end{aligned} \tag{5}$$

Where $rand_j$, $j \in \{5,6\}$ are uniformly distributed random numbers in the interval *(0, 1]*.

Thus the new *F* takes values in the interval *(0, 0.5]* and the new Cr takes values in the interval $(0,0.1] \cup \{0.5\}$.

F_{g+1} and Cr_{g+1} are obtained every iteration. So, they influence the mutation, crossover and selection operations of every new particle. The classic DE has three control parameters that need to be adjusted by the user. Apparently, the proposed ACDE has even more parameters, but it should be noted that we have fixed values for F_l, F_u, F_0, Cr_l, Cr_0, P_F *and* P_{Cr} for all the test problems in our ACDE algorithm.

V. Experimental Settings and Numerical Results

In order to make a fair comparison of DE and all the proposed algorithms, we fixed the same seed for random number generation so that the initial population is same for both the algorithms. The population size is taken as 50 for all the test problems for both the algorithms. However, this is a heuristic choice and may be increased, depending on the complexity of the problem. The control parameters, crossover rate and scaling factor F, for classical DE are fixed at 0.2 and 0.5 respectively. For each algorithm, the maximum number of iterations allowed was set to 5000 and the error goal was set as 1*e-04. A total of 30 runs for each experimental setting were conducted and the average fitness of the best solutions throughout the run was recorded.

In order to check the compatibility of the proposed ACDE algorithm we have tested it on a suite of ten benchmark problems; the mathematical models of the test problems with the true optimum value are given in Table I. The test bed comprises of a variety of problems ranging from a simple spherical function to highly multimodal functions with several local and global optima. We have also considered a noisy function, f_5, having a uniformly distributed random noise. In such type of functions the position of optima keeps changing and therefore it becomes challenging for a global optimization algorithm to locate the correct solution. Figure 1 shows the comparison of DE and ACDE based on number of function evaluations. The comparison of DE and ACDE based on convergence time is given in Figure 2. The performance curves of proposed DE algorithms with classical DE for selected benchmark problems are shown in Figures 3 – 6.

Performance comparisons of ACDE algorithm is performed with classical DE on the basis of standard performance measures like average fitness function value, Number of Function evaluations (NFE), CPU time and success rate (SR). From the numerical results given in Table II, we can see that ACDE algorithm gave better performance than classical DE in all the test cases except for the function f_4. For the function f_1, the difference in the average fitness function values for DE and ACDE is quite visible. The true global minimum for the function f_1 is located at 0.0. None of the algorithms were able to reach this value. However ACDE gave the value near about the true optimum and it is much better value in comparison to DE. In this case the improvement of ACDE in terms of fitness function value in comparison with DE is 99.55%. Similarly for function f_5, the use of adaptive control parameters improves the function value up to 80.5%. For other functions also, the proposed ACDE algorithm outperform the classical DE algorithm. If we compare the performance of ACDE with DE in terms of NFE then the improvement of ACDE algorithm is around 53%. From the numerical results of comparison of CPU time, it is clear that the proposed ACDE converges much faster than the classical DE; there is an improvement of 83% in CPU time. The total time taken by classical DE is 370.6 seconds whereas the total time taken by ACDE is 61.35 seconds only. If we talk about the success rate, which is also given in Table 2, we can see that on an average the proposed ACDE gives more than 88% success for all the test problems considered in this study.

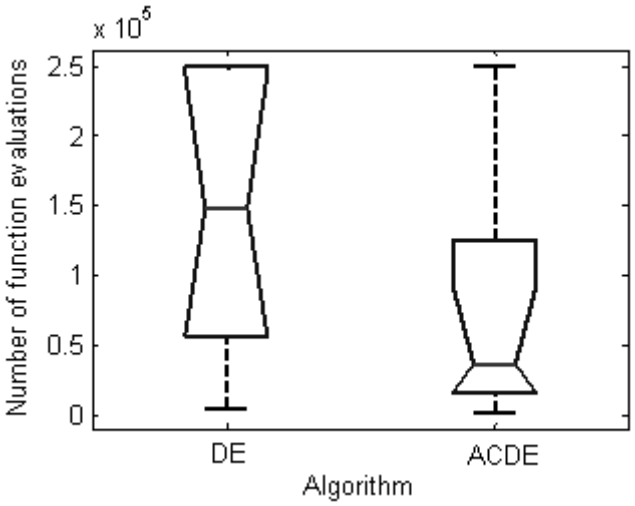

Figure 1. Comparison of DE and ACDE based on number of function evaluations of all test problems

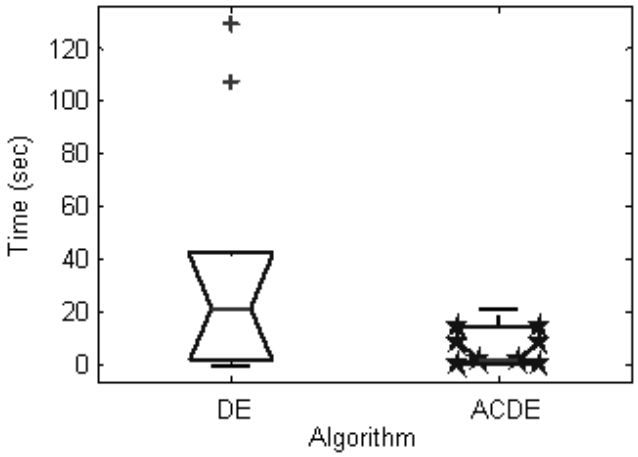

Figure 2. Comparison of DE and ACDE based on convergence time of all test problems

TABLE I. NUMERICAL BENCHMARK PROBLEMS

Function	Function Definition	Dimension	Range	Optimum
Rastringin Function	$f_1(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	30	[-5.12,5.12]	0
Spherical Function	$f_2(x) = \sum_{i=1}^{n} x_i^2$	30	[-5.12,5.12]	0
Griewank Function	$f_3(x) = \frac{1}{4000}\sum_{i=0}^{n-1} x_i^2 + \sum_{i=0}^{n-1}\cos(\frac{x_i}{\sqrt{i+1}}) + 1$	30	[-600,600]	0
Rosenbrock Function	$f_4(x) = \sum_{i=0}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	30	[-30,30]	0
Noisy Function	$f_5(x) = (\sum_{i=0}^{n-1}(i+1)x_i^4) + rand[0,1]$	30	[-1.28,1.28]	0
Schwefel Function	$f_6(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[-500,500]	-12569.5
Ackley Function	$f_7(x) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$	30	[-32,32]	0
Michalewicz function	$f_8(x) = -\sum_{i=1}^{n}\sin(x_i)(\sin(i\frac{x_i^2}{\pi}))^{2m}$, $m = 10$	30	[-π,π]	---
Himmelblau Function	$f_9(x) = (x_2 + x_1^2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + x_1$	2	[-5,5]	-3.78396
Shubert Function	$f_{10}(x) = \sum_{j=1}^{5} j\cos((j+1)x_1 + j)\sum_{j=1}^{5} j\cos((j+1)x_2 + j)$	2	[-10,10]	-186.7309

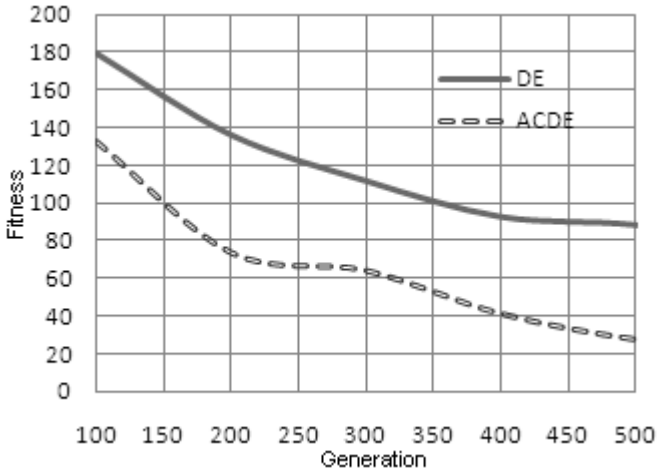

Figure 3. Performance curves of DE and ACDE for function f_1

TABLE II. NUMERICAL RESULTS OF DE AND ACDE ALGORITHMS

Function	Fitness value (Std)		Average NFE		CPU Time (sec)		Success Rate	
	DE	ACDE	DE	ACDE	DE	ACDE	DE	ACDE
f_1	29.9076 (1.34989)	0.132725 (0.338219)	250050	55676	42.8	3.8	-	100
f_2	6.87e-05 (9.13e-06)	5.82e-05 (1.10e-05)	57000	17590	8.6	1.2	100	100
f_3	7.70e-05 (8.63e-06)	6.26e-05 (1.76e-05)	175570	26120	28.9	1.8	100	100
f_4	26.3194 (1.4247)	34.3654 (18.1468)	250050	125320	106.9	21.2	-	80
f_5	0.0177813 (0.0042194)	0.003460 (0.000695)	250050	250050	37.9	16.1	-	-
f_6	-12474.7 (4.73753)	-12530 (70.623)	122525	44786	2.3	0.46	100	100
f_7	0.0001830 (2.077e-05)	0.000172 (3.23e-05)	100655	30526	13.9	2.2	100	100
f_8	-27.095 (0.32179)	-29.6199 (0.015583)	250050	125697	129.1	14.4	-	100
f_9	-3.28972 (0.388473)	-3.39549 (0.475781)	5470	3256	0.1	0.13	100	100
f_{10}	-186.731 (1.11e-07)	-186.731 (8.24e-08)	18120	9553	0.1	0.06	70	100
$\sum$			1479540	688574	370.6	61.35	570	880
Average			147954	68857.4	37.06	6.135	57	88
Improvement (%)				53.4602		83.4457		54.3859

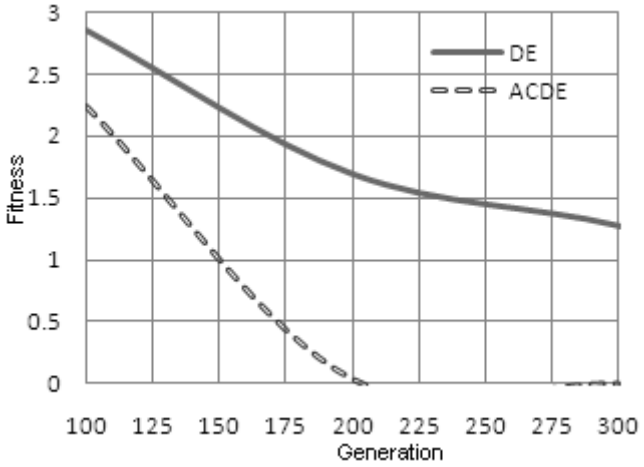

Figure 4. Performance curves of DE and ACDE for function f_2

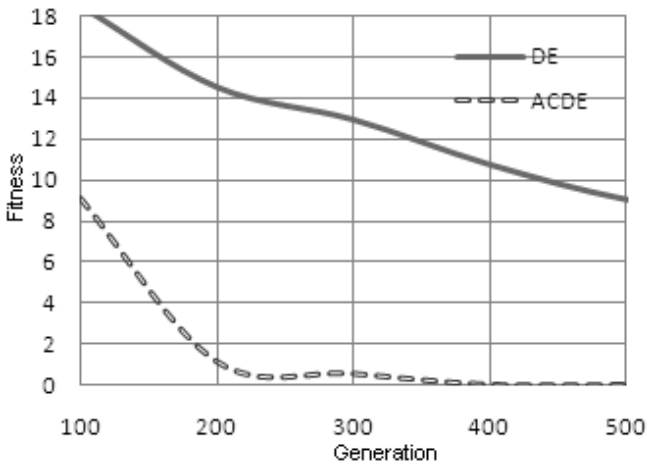

Figure 5. Performance curves of DE and ACDE for function f_3

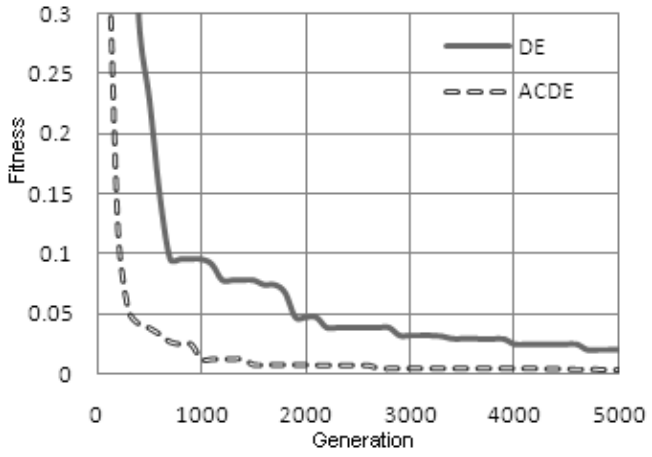

Figure 6. Performance curves of DE and ACDE for function f_5

VI. CONCLUSION

In this paper, a new Differential Evolution based on adaptive control parameters called ACDE was proposed. ACDE algorithm uses a few simple rules to design adaptive scaling factor F and crossover rate C_r. The ACDE algorithm was tested on a test bed of ten standard benchmark problems and the results were compared with the classical DE in terms of average fitness function value, number function evaluations, convergence time and success rate. From the numerical results, we can see that the proposed ACDE algorithm helps in improving the solution quality as well as the convergence rate in all the test problems. In future we will extend the proposed ACDE algorithm for solving more comprehensive set of test problems including real life problems. Also we plan to compare the proposed algorithm with other models of DE.

REFERENCES

[1] Storn, R., "System design by constraint adaptation and differential evolution", IEEE Transactions on Evolutionary Computation, Vol. 3, 1999, pp. 22-34.

[2] Storn, R. and Price, K., "Differential Evolution – a simple and efficient Heuristic for global optimization over continuous spaces", Journal Global Optimization, Vol. 11, 1997, pp. 341 – 359.

[3] Price, K. and Storn, R., "Differential Evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report, International Computer Science Institute, Berkley, 1995.

[4] Engelbrecht, A. P., "Fundamentals of Computational Swarm Intelligence", John Wiley & Sons Ltd, 2005.

[5] Lampinen, J. and Zelinka, I., "On stagnation of the Differential Evolution Algorithm", In: Pavel Ošmera, (ed.) Proc. of MENDEL 2000, 6th International Mendel Conference on Soft Computing, 2000, pp. 76 – 83.

[6] Gämperle, R., Müller, S. D. and Koumoutsakos, P., "A Parameter Study for Differential Evolution", WSEAS NNA-FSFS-EC 2002. Interlaken, Switzerland, WSEAS, 2002, pp. 293 – 298.

[7] Liu, J. and Lampinen, J., "On Setting the Control Parameter of the Differential Evolution Method", In Proc. of 8th Int. Conf. Soft Computing *(MENDEL 2002)*, 2002, pp. 11–18.

[8] Liu, J. and Lampinen, J., "A Fuzzy Adaptive Differential Evolution Algorithm", Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 9(6), 2005, pp. 448 – 462.

[9] Das, S., Konar, A. and Chakraborty, U. K., "Two improved differential evolution schemes for faster global search", ACM-SIGEVO Proceedings of GECCO, Washington D.C., 2005, pp. 991-998.

[10] Teo, J., "Exploring Dynamic Self-adaptive Populations in Differential Evolution", Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 10 (8), 2006, pp. 673 – 686.

[11] Brest, J., Boˇskovi´c, B., Greiner, S., ˇZ umer, V. and Mauˇcec, M. S., "Performance Comparison of Self-Adaptive and Adaptive Differential Evolution Algorithms", Technical Report #2006-1-LABRAJ, University of Maribor, Faculty of Electrical Engineering and Computer Science, Slovenia, 2006, http://marcel.uni-mb.si/janez/brest-TR1.html.

[12] Brest, J., Greiner, S., Boˇskovi´c, B., Mernik, M. and ˇZumer, V., "Self- Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems", IEEE Transactions on Evolutionary Computation, Vol. 10(6), 2006, pp. 646 – 657.

[13] Ali, M. M., "Differential Evolution with Preferential Crossover", European Journal of Operational Research, Vol. 181, 2007, pp. 1137 – 1147.

[14] Yang, Z., Tang, K. and Yao, X., "Self-adaptive Differential Evolution with Neighborhood Search", In Proc. IEEE Congress on Evolutionary Computation, Hong Kong, 2008, pp. 1110-1116.

[15] Qin, A. K., Huang, V. L. and Suganthan, P. N., "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization", IEEE Transactions on Evolutionary Computations, Vol. 13 (2), 2009, pp. 398 – 417.

[16] Chakraborty, U. K., "Advances in Differential Evolution", (Ed.) Springer-Verlag, Heidelberg, 2008.