CONTROL PARAMETERS IN SELF-ADAPTIVE DIFFERENTIAL EVOLUTION

Janez Brest, Viljem Žumer, Mirjam Sepesy Maučec
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
{janez.brest,zumer,mirjam.sepesy}@uni-mb.si

Abstract In this paper we present experimental results to show deep view on how self-adaptive mechanism works in differential evolution algorithm. The results of the self-adaptive differential evolution algorithm were evaluated on the set of 24 benchmark functions provided for the CEC2006 special session on constrained real parameter optimization. In this paper we especially focus on how the control parameters are being changed during the evolutionary process.

Keywords: Control parameters, Differential evolution, Self-adapting

1. Introduction

Differential Evolution (DE) [8, 9, 10, 13, 14, 15, 16] has been shown to be a powerful evolutionary algorithm for global optimization in many real problems [11, 12]. Although the DE algorithm has been shown to be a simple yet powerful evolutionary algorithm for optimizing continuous functions, users are still faced with the problem of preliminary testing and hand-tuning of the evolutionary parameters prior to commencing the actual optimization process [16].

Different problems usually require different setting for the control parameters. Self-adaptation allows an evolution strategy to adapt itself to any general class of problems by reconfiguring itself accordingly, and to do this without any user interaction [1, 2, 6]. Based on the experiment in [4], the necessity of changing control parameters during the optimization process is also confirmed. In literature, self-adaptation is usually applied to the F and CR control parameters [3, 4].

In our previous paper [5] the performance of the self-adaptive differential evolution algorithm was evaluated on the set of 24 benchmark functions provided for the CEC2006 special session on constrained real parameter optimization [7]. The method in [5] extended individuals that have not only decision variables but also control parameters F and CR, where F is a scaling factor

and CR is a crossover rate. These parameters are changed/optimized by DE, too. The authors utilized the lexicographic ordering, in which the constraint violation precedes the objective function, to solve constrained problems.

In this paper we investigate how these parameters adapt during search for some of the test functions (i.e. some typical runs). Do they really change much and how?

The main focus in this paper is related with our previous paper [5] where the performance of the self-adaptive differential evolution algorithm was evaluated on the set of 24 benchmark functions [7]. In [5] results are presented, how efficient our self-adaptive DE algorithm is on constraint-base optimization. In this paper we focus only on a self-adapting control parameters. We want to answer, how the control parameter are being changed during the evolutionary process.

2. Background

In this section we give overview of previous works, which gives the basis of this paper. The original differential evolution (DE) algorithm is briefly presented. Then the self-adaptive mechanism used in our DE algorithm is outlined.

2.1 The Differential Evolution Algorithm

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 parameters: amplification factor of the difference vector, F, crossover control parameter, CR, and population size, NP.

The population of the original DE algorithm [13, 14, 15] contains NP D-dimensional vectors:

$$\vec{x}_{i,G} = \{x_{i,1,G}, x_{i,2,G}, \ldots, x_{i,D,G}\},\ i = 1, 2, \ldots, NP.$$

G denotes the generation. During one generation for each vector, DE employs the mutation and crossover operations to produce a trial vector:

$$\vec{u}_{i,G} = \{u_{i,1,G}, u_{i,2,G}, \ldots, u_{i,D,G}\},\ i = 1, 2, \ldots, NP.$$

Then a selection operation is used to choose vectors for the next generation $(G + 1)$.

The initial population is selected randomly in a uniform manner 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. After initialization, DE performs several vector transforms (operations) in a process called evolution.

2.2 Mutation Operation

Mutation for each population vector creates a mutant vector:

$$\vec{x}_{i,G} \Rightarrow \vec{v}_{i,G} = \{v_{i,1,G}, v_{i,2,G}, \ldots, v_{i,D,G}\},\ i = 1, 2, \ldots, NP.$$

Mutant vector can be created by using one of the mutation strategies. There are many original DE strategies. The strategies used in this paper are:

- 'rand/1': $\vec{v}_{i,G} = \vec{x}_{r_1,G} + F \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G})$,
- 'current to best/1':
  $\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{best,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$,
- 'rand/2':
  $\vec{v}_{i,G} = \vec{x}_{r_1,G} + F \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}) + F \cdot (\vec{x}_{r_4,G} - \vec{x}_{r_5,G})$,

where the indexes r_1, r_2, r_3, r_4, r_5 represent the random and mutually different integers generated within range $[1, NP]$ and also different from index i. F is a mutation scale factor within the range $[0, 2]$, usually less than 1. $\vec{x}_{best,G}$ is the best vector in generation G.

2.3 Crossover Operation

After mutation, a 'binary' crossover operation forms the final trial vector, according to the i-th population vector and its corresponding mutant vector.

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j,G} & \text{otherwise} \end{cases}$$

$$i = 1, 2, \ldots, NP \text{ and } j = 1, 2, \ldots, D.$$

CR is a crossover parameter or factor within the range $[0, 1)$ and presents the probability of creating parameters for trial vector from a mutant vector. Index j_{rand} is a randomly chosen integer within the range $[1, NP]$. It is responsible for the trial vector containing at least one parameter from the mutant vector.

2.4 Selection Operation

The selection operation selects according to the fitness value of the population vector and its corresponding trial vector, which vector will survive to be a member of the next generation. For example, if we have a minimization problem, we will use the following selection rule:

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if } f(\vec{u}_{i,G}) < f(\vec{x}_{i,G}), \\ \vec{x}_{i,G} & \text{otherwise.} \end{cases}$$

2.5 The Self-Adaptive Differential Evolution Algorithm

In [4] a self-adaptive control mechanism was used to change the control parameters F and CR during the run.

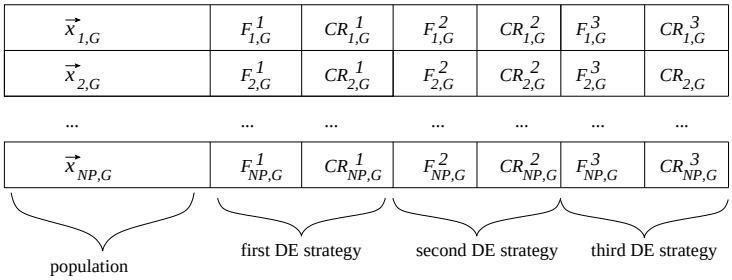

Figure 1. Self-adapting: encoding aspect of three DE strategies.

Figure 1 shows a solution how the control parameters of three original DE's strategies are encoded in each individual. Each strategy uses its own control parameters. The solution to apply even more strategies into our algorithm is straight-forward. New control parameters $F^k_{i,G+1}$ and $CR^k_{i,G+1}$, $k = 1, 2, 3$, were calculated as follows:

$$F^k_{i,G+1} = \begin{cases} F_l + rand_1 * F_u & \text{if } rand_2 < \tau_1, \\ F_{i,G} & \text{otherwise,} \end{cases}$$

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

and they produce control parameters F and CR in a new parent vector. k represents selected DE strategy. When a new parent vector is calculated, only one strategy is active. In each iteration one strategy is chosen to be active. $rand_j, j \in \{1, 2, 3, 4\}$ are uniform random values within the range $[0, 1]$. τ_1 and τ_2 represent probabilities to adjust 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 $[0.1, 1.0]$, and the new CR from $[0, 1]$ in a random manner. $F_{i,G+1}$ 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 $\vec{x}_{i,G+1}$.

In experiments in [5], the proposed jDE-2 algorithm uses the following three strategies 'rand/1/bin', 'current to best/1/bin', and 'rand2/bin'. The first pair of self-adaptive control parameters F and CR belongs to the 'rand/1/bin' strategy and the second pair belongs to 'current to best/1/bin' strategy, etc. The population size NP was set to 200. The maximal number of function evaluations (FES) is 500,000 for all benchmark functions.

The algorithm distinguishes between feasible and infeasible individuals: any feasible solution is better than any infeasible one.

The jDE-2 algorithm was tested on 24 CEC2006 special session benchmark functions. For 22 functions the jDE-2 algorithm has successfully found feasible solution. For $g20$ and $g22$ functions no feasible solution has been found.

3. Experimental Results

In this section we present results of experiments, which were made in order to find an answer, how the control parameters are adapted during evolutionary process.

In self-adaptive DE, F and CR values are being changed during evolutionary process. If we want to look into evolutionary process, we should look at fitness values.

In Figures 2–4 F and CR values of the active strategy are depicted for the selected set of functions $g01$, $g02$, $g05$, $g07$, $g10$, $g14$, $g15$, $g16$, $g17$, $g18$, $g19$, $g20$. A dot is plotted only when the best fitness value in generation is improved.

The values of control parameter F and CR for function $g01$ are quite equally distributed, F takes value from the $[0.1, 1]$ and CR from the $[0, 1]$.

For function $g02$ the values of control parameter F are in most cases less or equal 0.5 in the first 200,000 evaluations. After that F values are predominantly greater than 0.5. The values of control parameter CR are near 1 mostly.

Sometimes algorithm solves test problem before reaching the maximal number of FES, therefore some graphs (e.g., functions $g01$, $g10$, etc.) have not dots for all FES.

It can be seen that the graphs differ from each other to a great extend. It is difficult to obtain (general) one set of control parameter values, which will fit the best for all benchmark problems.

In the additional experiment, we run our algorithm without self-adaptation. The values of control parameters were $F = 0.5$ and $CR = 0.9$, and they were fixed during evolutionary process.

The algorithm with self-adaptation performs 11 % better than algorithm with fixed control parameters. The detailed performance results of our self-adaptive algorithm are in [5].

4. Conclusions

This paper shows that the DE control parameters F and CR changed (adapted) their values during evolutionary process. For selected CEC2006 benchmark functions the graphs of F and CR values during the evolution process are presented in the paper.

The experimental results confirm the hypothesis that the best setting for control parameters is problem dependent.

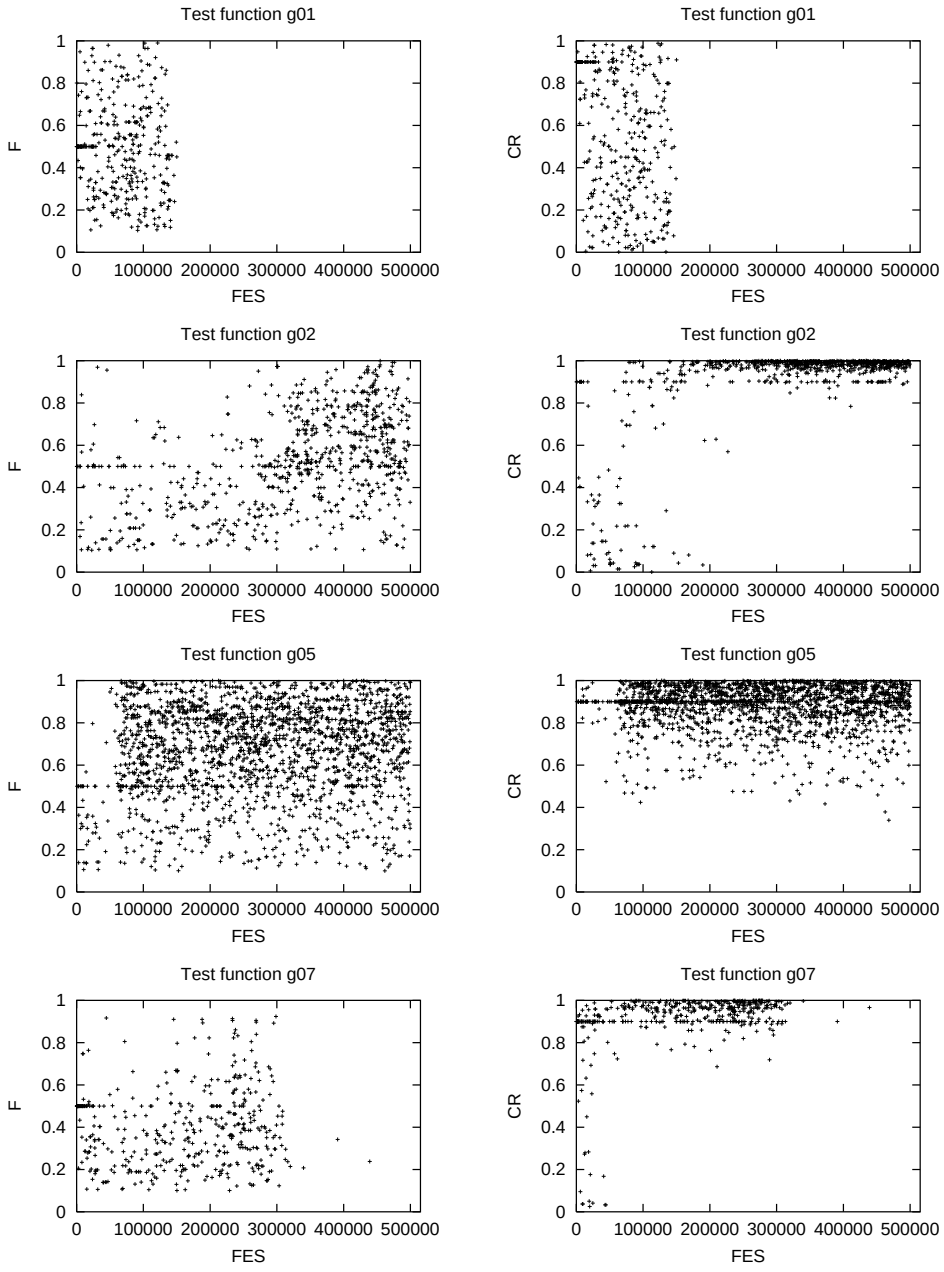

Figure 2. F and CR values for functions $g01$, $g02$, $g05$, and $g07$.

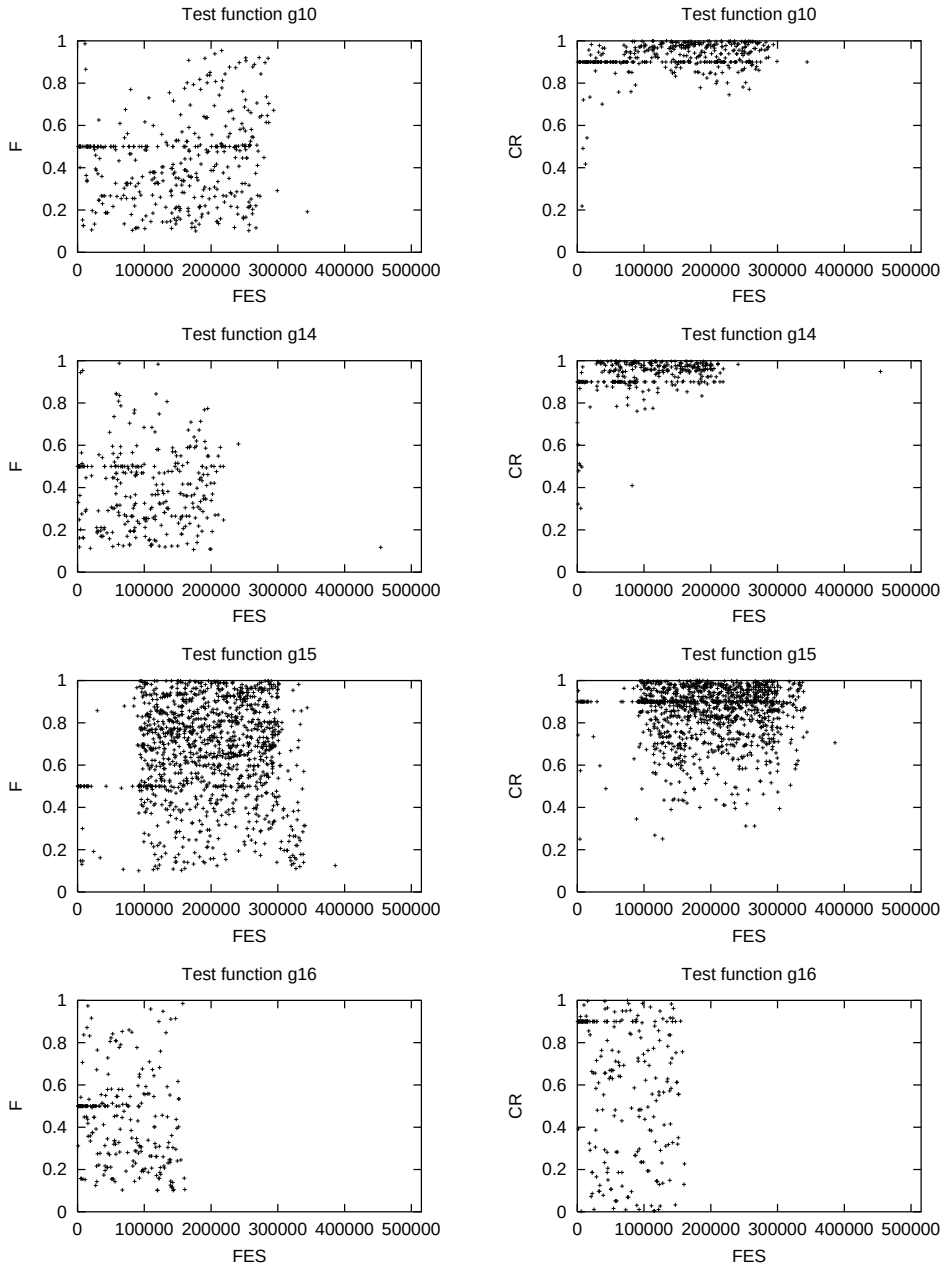

Figure 3. F and CR values for functions $g10$, $g14$, $g15$, and $g16$.

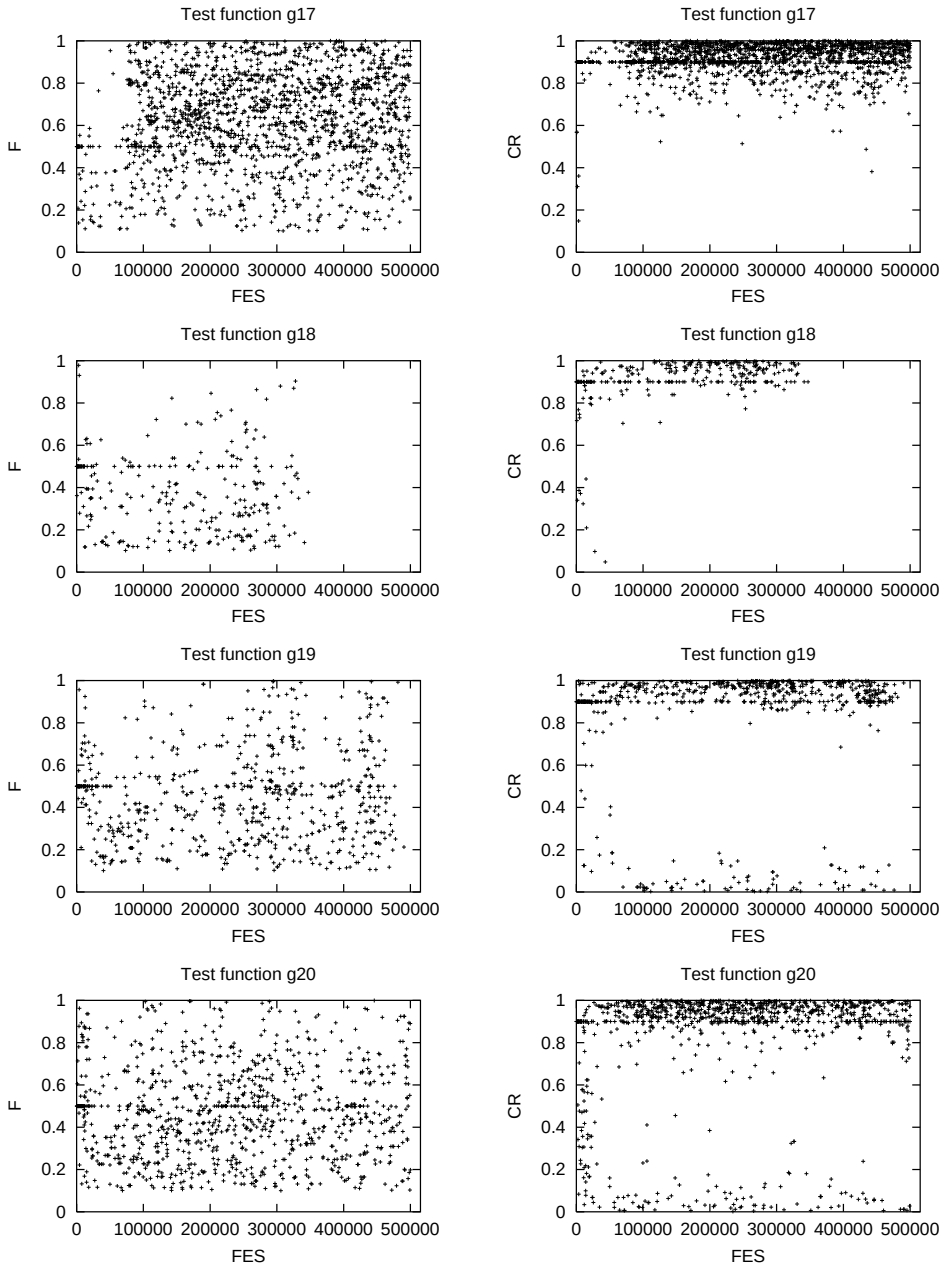

Figure 4. F and CR values for functions $g17$, $g18$, $g19$, and $g20$.

In this paper we used three DE strategies. The analysis how the control parameters are changed in particularly DE strategy is a challenge for the future work.

References

[1] T. Bäck. Adaptive Business Intelligence Based on Evolution Strategies: Some Application Examples of Self-Adaptive Software. *Infor. Sc.*, 148(1-4):113–121, 2002.

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.). *Handbook of Evolutionary Computation*. Oxford University Press, New York and Institute of Physics Publishing, Bristol, 1997.

[3] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. Sepesy Maučec. Performance Comparison of Self-Adaptive and Adaptive Differential Evolution Algorithms. *Soft Comput.*, 2006. To appear.

[4] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Trans. Evol. Comput.*, 2006. To appear.

[5] J. Brest, V. Žumer, and M. Sepesy Maučec. Self-adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 215–222, Vancouver, BC, Canada, 2006.

[6] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing. Springer-Verlag, Berlin, 2003.

[7] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, N. Suganthan, C. A. C. Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Report #2006005, Nanyang Technological University, Singapore, 2005. `www.ntu.edu.sg/home/EPNSugan`.

[8] J. Liu and J. Lampinen. Adaptive Parameter Control of Differential Evolution. In *Proc. 8th International Conference on Soft Computing*, pages 19–26, Brno, Czech Republic, 2002.

[9] J. Liu and J. Lampinen. On Setting the Control Parameter of the Differential Evolution Method. In *Proc. 8th International Conference on Soft Computing*, pages 11–18, Brno, Czech Republic, 2002.

[10] J. Liu and J. Lampinen. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Comput.*, 9(6):448–462, 2005.

[11] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[12] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evol. Comput.*, 4(1):1–32, 1996.

[13] J. Rönkkönen, S. Kukkonen, and K. V. Price. Real-Parameter Optimization with Differential Evolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2005)*, vol. 1, pages 506–513, Edinburg, UK, 2005.

[14] R. Storn and K. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

[15] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Global Opt.*, 11(4):341–359, 1997.

[16] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.*, 10(8):673–686, 2006.