Proceedings of the International Multiconference on Computer Science and Information Technology pp. 207–213
ISSN 1896-7094

Differential Evolution: Competitive Setting of Control Parameters*

Josef Tvrdík

University of Ostrava, Department of Computer Science
30. Dubna 22, 701 03 Ostrava, Czech Republic
Josef.Tvrdik@osu.cz
http://albert.osu.cz/tvrdik

Abstract. This paper is focused on the adaptation of control parameters in differential evolution. The competition of different control parameter settings was proposed in order to ensure the self-adaptation of parameters within the search process. Several variants of such algorithm were tested on six functions at four levels of search-space dimension. This competitive differential evolution proved to be more reliable and less time-consuming than the standard differential evolution. The competitive differential evolution also outperformed other algorithms that were tested.

1 Introduction

We will deal with the global optimization problem: for a given objective function

$$f : D \to \mathbb{R}, \quad D \subset \mathbb{R}^d,$$

the point $\boldsymbol{x}^*$ is to be found such that $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in D} f(\boldsymbol{x})$. The point $\boldsymbol{x}^*$ is called the global minimum point and D is the search space. We focus on the problems, where the objective function is continuous and the search space is closed compact set, $D = \prod_{i=1}^{d}[a_i, b_i],\ a_i < b_i,\ i = 1, 2, \ldots, d$ (box constrains).

The problem of the global optimization is hard and plenty of stochastic algorithms were proposed for its solution, see e.g. [2, 11]. The authors of many such stochastic algorithms claim the efficiency and the reliability of searching for the global minimum. The reliability means that the point with minimal function value found in the search process is sufficiently close to the global minimum point and the efficiency means that the algorithm finds a point sufficiently close to the global minimum point at reasonable time. However, when we use such algorithms, we face the problem of the setting their control parameters. The efficiency and the reliability of many algorithms is strongly dependent on the values of control parameters. Recommendations given by authors are often vague or uncertain, see e.g. [12, 22]. A user is supposed to be able to change the parameter values according to the results of trial-and-error preliminary experiments with the search process. Such attempt is not acceptable in tasks, where the global optimization is one step on the way to the solution of the user's problem or when the user has no experience in fine art of control parameter tuning.

Adaptive robust algorithms reliable enough at reasonable time-consumption without the necessity of fine tuning their input parameters have been studied in recent years. The proposal of an adaptive generator of robust algorithms is described in Deb [4]. Winter et al. [17] proposed a flexible evolutionary agent for real-coded genetic algorithms. Theoretical analysis done by Wolpert and Macready implies, that any search algorithm cannot outperform the others for all objective functions [18]. In spite of this fact, there is empirical evidence, that some algorithms can outperform others for relatively wide class of problems both in the convergence rate and in the reliability of finding the global minimum point. Thus, the way to the adaptive algorithms leads rather trough the experimental research than trough purely theoretical approach.

2 Differential Evolution and its Control Parameters

The differential evolution (DE) works with two population P and Q of the same size N. The algorithm in pseudo-code is written as Algorithm 1. A new trial point $\boldsymbol{y}$ is composed of the current point $\boldsymbol{x_i}$ of old population and the point $\boldsymbol{u}$ obtained by using mutation. If $f(\boldsymbol{y}) < f(\boldsymbol{x_i})$ the point $\boldsymbol{y}$ is inserted into the new population Q instead of $\boldsymbol{x_i}$. After completion of the new population Q the old population P is replaced by Q and the search continues until stopping condition is fulfilled.

* This work was supported by the grant 201/05/0284 of the Czech Grant Agency and by the research scheme MSM 6198898701 of the Institute for Research and Applications of Fuzzy Modeling.

Algorithm 1. Differential evolution

```
1   generate P = (x_1, x_2, ..., x_N); (N points in D)
2   repeat
3       for i := 1 to N do
4           compute a mutant vector u;
5           create y by the crossover of u and x_i;
6           if f(y) < f(x_i) then insert y into Q
7                            else insert x_i into Q
8           endif;
9       endfor;
10      P := Q;
11  until stopping condition;
```

There are several variants how to generate the mutant point $\boldsymbol{u}$. One of the most popular (called DER in this text) generates the point $\boldsymbol{u}$ by adding the weighted difference of two points

$$\boldsymbol{u} = \boldsymbol{r}_1 + F\,(\boldsymbol{r}_2 - \boldsymbol{r}_3)\,, \tag{1}$$

where $\boldsymbol{r}_1, \boldsymbol{r}_2$ and $\boldsymbol{r}_3$ are three distinct points taken randomly from P (not coinciding with the current $\boldsymbol{x}_i$) and $F > 0$ is an input parameter. Another variant called DEBEST generates the point $\boldsymbol{u}$ according to formula

$$\boldsymbol{u} = \boldsymbol{x}_{\min} + F\,(\boldsymbol{r}_1 + \boldsymbol{r}_2 - \boldsymbol{r}_3 - \boldsymbol{r}_4)\,, \tag{2}$$

where $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4$ are four distinct points taken randomly from P (not coinciding with the current $\boldsymbol{x}_i$), $\boldsymbol{x}_{\min}$ is the point of P with minimal function value, and $F > 0$ is an input parameter.

The elements y_j, $j = 1, 2. \ldots, d$ of trial point $\boldsymbol{y}$ are built up by the crossover of its parents $\boldsymbol{x}_i$ and $\boldsymbol{u}$ using the following rule

$$y_j = \begin{cases} u_j & \text{if} \quad U_j \le C \quad \text{or} \quad j = l \\ x_{ij} & \text{if} \quad U_j > C \quad \text{and} \quad j \ne l\,, \end{cases} \tag{3}$$

where l is a randomly chosen integer from $\{1, 2, \ldots, d\}$, $U_1, U_2, \ldots, U_d$ are independent random variables uniformly distributed in $[0,\ 1)$, and $C \in [0, 1]$ is an input parameter influencing the number of elements to be exchanged by crossover. Eq. (3) ensures that at least one element of $\boldsymbol{x}_i$ is changed even if $C = 0$.

The differential evolution has become one of the most popular algorithms for the continuous global optimization problems in recent years, see [5]. But it is known that the efficiency of the search for the global minimum is very sensitive to the setting of values F and C. The recommended values are $F = 0.8$ and $C = 0.5$, but even Storn and Price in their principal paper [12] use $0.5 \le F \le 1$ and $0 \le C \le 1$ depending on the results of preliminary tuning. They also set the size of population less than the recommended $N = 10\,d$ in many of their test tasks.

Many papers deal with the setting of control parameters for differential evolution. Ali and Törn [1] suggested to adapt the value of the scaling factor F within the search process according to the equation

$$F = \begin{cases} \max(F_{\min}, 1 - |\frac{f_{\max}}{f_{\min}}|) & \text{if } |\frac{f_{\max}}{f_{\min}}| < 1 \\ \max(F_{\min}, 1 - |\frac{f_{\min}}{f_{\max}}|) & \text{otherwise}\,, \end{cases} \tag{4}$$

where $f_{\min}$, $f_{\max}$ are respectively the minimum and maximum function values in the population and $F_{\min}$ is an input parameter ensuring $F \in [F_{\min}, 1]$. According to [1] this calculation of F reflects the demand to make the search more diversified at early stage and more intensified at latter stages, i.e. to produce rather larger values of F for large difference $f_{\max} - f_{\min}$, and rather smaller values of F otherwise. The rule (4) works properly only for $f_{\min} > 0$. When $f_{\max} > 0$ and $f_{\min} < 0$, especially if $|f_{\max}| < |f_{\min}|$, the values of F fluctuate very rapidly in $[F_{\min}, 1]$ even if the changes in $f_{\max}$ or $f_{\min}$ are small. However, from practical point of view, it occurs only as a short episode of the search process in most optimization tasks. Moreover, the values of F calculated by using (4) are not invariant to the shift of the objective function f values by a constant, although such a shift should not affect the search process, because the shape of the is the the function does change by the shift. In spite of the facts we can take the proposal (4) as an acceptable trade-off between its simplicity and performance.

Zaharie [19] derived the critical interval for the control parameters of DE. This interval ensures to keep the mean of population variance non-decreasing, which results in the following relationship

$$2\,p\,F^2 - \frac{2p}{N} + \frac{p^2}{N} > 0\,, \tag{5}$$

where $p = \max(1/d, C)$ is the probability of "differential perturbation" according to (3). The relationship (5) implies that the mean of population variance is non-decreasing, if $F > \sqrt{1/N}$, but practical reason of such result is very limited, because it brings no new information when we compare this result with the minimal value of $F = 0.5$ used in [12] and in other applications of differential evolution.

Some other attempts to the adaptation of DE control parameters have appeared, see e.g. [6], [7], [13], and [20]. Recent state of adaptive parameter control in differential evolution is summarized by Liu and Lampinen [8].

New idea of self-adaptation of control parameters F and C in differential evolution was presented by Brest et al. [3]. The values of F and C can be changed in each generation with probability τ_1, τ_2 resp. New values of F are distributed uniformly in $[F_l, F_u]$ and new C are also uniform random values $\in [0, 1]$.

3 Competition in Differential Evolution

The setting of the control parameters can be made adaptive trough the implementation of a competition into the algorithm. This idea similar to the competition of local-search heuristics in evolutionary algorithm [14] or in controlled random search [15] was proposed this year [16].

Let us have H settings (different values of F and C used in the statements on line 4 and 5 of Algorithm 1) and choose among them at random with the probability $q_h,\ h = 1, 2, \ldots, H$. The probabilities can be changed according to the success rate of the setting in preceding steps of search process. The h-th setting is successful if it generates such a trial point $\mathbf{y}$ that $f(\mathbf{y}) < f(\mathbf{x}_i)$. When n_h is the current number of the h-th setting successes, the probability q_h can be evaluated simply as the relative frequency

$$q_h = \frac{n_h + n_0}{\sum_{j=1}^{H}(n_j + n_0)}\ , \tag{6}$$

where $n_0 > 0$ is a constant. The setting of $n_0 \geq 1$ prevents a dramatic change in q_h by one random successful use of the h-th parameter setting. In order to avoid the degeneration of process the current values of q_h are reset to their starting values ($q_h = 1/H$) if any probability q_h decreases bellow a given limit $\delta > 0$.

It is supposed that such a competition of different settings will prefer successful settings. The competition will serve as an self-adaptive mechanism of setting control parameter appropriate to the problem actually solved.

4 Experiments and Results

Four variants of such competitive differential evolution were implemented and tested. Three values of control parameter C were used in all the variants, namely $C = 0$, $C = 0.5$, and $C = 1$.

- DER9 – the mutant vector $\boldsymbol{u}$ is generated according to (1), nine settings of control parameters are all the combinations of three F-values ($F = 0.5$, $F = 0.8$, and $F = 1$) with three values of C given above,
- DEBEST9 – the mutant vector $\boldsymbol{u}$ is generated according to (2), nine settings of control parameters F and C like in DER9,
- DERADP3 – the mutant vector $\boldsymbol{u}$ is generated according to (1), F adaptive according to (4), three settings of C.
- DEBR18 – 18 settings, aggregation of settings used in DER9 and DEBEST9, implemented due to the good performance of DER9 and DEBEST9 in the test tasks.

Population size was set to $N = \max(20, 2\,d)$, and parameters for competition control were set to $n_0 = 2$, and $\delta = 1/(5\,H)$ in all the test tasks.

These variants of competitive differential evolution were compared with four other algorithms. One of them was the standard DER with recommended values $F = 0.8$ and $C = 0.5$, and population size the same as in the competitive variants of differential evolution. The second algorithm was self-adaptive differential evolution (SADE) proposed by Brest et al. with values of its control parameters recommended in [3], i.e. population size $N = 10\,d$, input parameters for self-adaptation $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$, and $F_u = 0.9$. The third algorithm was the controlled random search with eight competing local-search heuristics (CRS8HC), described in [15] including the setting of its control parameters. The last algorithm was SOMA (all-to-one

variant), see [21] and [22]. Its control parameters were set to values $N = 5\,d$, *mass*=3, *step*=0.11, and *prt*=0.1 as it was recommended by Ivan Zelinka in private communication.

The search for the global minimum was stopped if $f_{\max} - f_{\min} < 1E-07$ or the number of objective function evaluations exceeds the input upper limit $20000\,d$.

The algorithms were tested on six functions commonly used in experimental tests. Two of them are unimodal (first De Jong, Rosenbrock), the other test functions are multimodal. Definitions of the test functions can be found e.g. in [12] or [1]. The search spaces D in all test tasks were symmetric and with the same interval in each dimension, $a_i = -b_i,\ b_i = b_j\ i, j = 1, 2, \ldots, d$. The values of b_i were set as follows: 2.048 for Rosenbrock function, 5.12 for first De Jong and Rastrigin function, 30 for Ackley function, 400 for Griewangk function and 500 for Schwefel function. The test were preformed for all the functions at four levels of dimension d of search spaces, namely $d = 2$, $d = 5$, $d = 10$ and $d = 30$. One hundred of repeated runs were carried out for each function and level of d.

Table 1. Differential evolution with competing parameter settings

Algorithm		DERB18				DER9				DEBEST9				DERADP3			
Function	d	λ_f	λ_m	ne	R	λ_f	λ_m	rne	R	λ_f	λ_m	rne	R	λ_f	λ_m	rne	R
ackley	2	7.1	6.8	2409	100	7.2	6.9	-9	100	7.1	6.7	10	100	7.2	6.8	3	100
dejong1	2	8.4	3.7	1162	100	8.4	3.7	-8	100	8.4	3.7	7	100	8.4	3.7	14	100
griewangk	2	8.5	3.5	2876	100	8.3	3.4	-12	100	8.5	3.5	21	100	8.0	3.2	7	93
rastrig	2	8.5	4.9	1778	100	8.4	4.9	-11	100	8.5	4.9	11	100	8.5	5.0	1	100
rosen	2	8.3	4.6	1956	100	8.2	4.5	-5	100	8.1	4.7	11	100	8.0	4.5	27	100
schwefel	2	7.5	5.5	1640	100	7.5	5.5	-7	100	7.5	5.5	8	100	7.5	5.5	-21	100
ackley	5	6.4	6.2	6401	100	6.5	6.2	-11	100	6.5	6.2	17	100	6.5	6.3	-6	100
dejong1	5	7.2	3.2	3176	100	7.2	3.2	-11	100	7.2	3.2	14	100	7.4	3.3	15	100
griewangk	5	7.2	2.5	8686	100	7.2	2.5	-15	99	7.2	2.6	40	100	6.5	2.3	7	85
rastrig	5	7.2	4.4	4989	100	7.2	4.4	-13	100	7.2	4.4	18	100	6.9	4.2	4	94
rosen	5	6.9	4.2	6256	100	6.7	4.1	47	97	6.8	4.2	14	99	2.1	1.4	128	30
schwefel	5	7.4	5.4	4564	98	7.4	5.4	-12	98	7.4	5.4	12	99	7.4	5.4	-28	99
ackley	10	6.1	5.9	13569	100	6.1	5.9	-15	100	6.1	5.9	24	100	5.5	5.4	-26	90
dejong1	10	6.7	3.0	6973	100	6.6	3.0	-14	100	6.7	3.1	22	100	6.8	3.1	7	100
griewangk	10	6.6	2.1	13153	99	6.6	2.1	-18	100	6.8	2.2	37	100	6.3	2.0	-10	91
rastrig	10	6.7	4.2	10711	100	6.7	4.2	-13	100	6.6	4.2	25	99	6.5	4.1	1	96
rosen	10	6.3	4.0	20524	100	5.8	3.5	110	95	6.4	4.2	15	100	0.0	0.0	66	0
schwefel	10	7.4	5.4	9964	99	7.3	5.3	-14	97	7.4	5.4	21	98	6.9	4.9	-33	90
ackley	30	5.9	5.8	142208	100	5.8	5.8	-13	100	6.0	5.9	21	100	5.8	5.8	-40	100
dejong1	30	6.4	3.0	78664	100	6.3	3.0	-13	100	6.5	3.1	21	100	6.4	3.1	-3	100
griewangk	30	6.4	1.6	103095	100	6.3	1.6	-13	100	6.5	1.7	24	100	6.4	1.7	-7	100
rastrig	30	6.4	4.1	110071	100	6.3	4.2	-12	100	6.5	4.2	25	100	6.4	4.2	-2	100
rosen	30	6.3	4.3	381972	100	6.2	4.2	1	100	6.4	4.3	28	100	0.0	0.0	57	0
schwefel	30	7.5	5.4	108050	100	7.5	5.4	-12	100	7.5	5.5	20	100	7.5	5.4	-37	100

The accuracy of the result obtained by the search for the global minimum was evaluated according to the number of duplicated digits when compared with the right certified result. The number of duplicated digits λ can be calculated via *log relative error* [10]:

- If $c \neq 0$, the λ is evaluated as

$$\lambda = \begin{cases} 0 & \text{if } \frac{|m-c|}{|c|} \geq 1 \\ 11 & \text{if } \frac{|m-c|}{|c|} < 1 \times 10^{-11} \\ -\log_{10}\left(\frac{|m-c|}{|c|}\right) & \text{otherwise}, \end{cases} \tag{7}$$

  where c denotes the right certified value and m denotes the value obtained by the search.
- If $c = 0$, the λ is evaluated as

$$\lambda = \begin{cases} 0 & \text{if } |m| \geq 1 \\ 11 & \text{if } |m| < 1 \times 10^{-11} \\ -\log_{10}(|\ m\ |) & \text{otherwise}. \end{cases} \tag{8}$$

Two values of the number of duplicated digits are reported in the results: λ_f for the function value, and λ_m, which is minimal λ for the global minimum point $(x_1, x_2, \ldots, x_d)$ found by the search.

The results of the competitive DE are presented in Table 1. The time consumption is expressed as the average number (ne) of the objective function evaluations needed to reach the stopping condition. Columns of Table 1 (and also Table 2) denoted rne contain the relative change of ne in percents when compared with DEBR18 in Table 1. Therefore, the negative values of rne mean less time consumption with respect to DEBR18, the positive values bigger one, the value $rne = 100$ means that ne is twice bigger, the value $rne = -50$ means half ne in comparison with DEBR18. The reliability of search is reported in the columns λ_f and λ_m, where are the average values of one hundred runs, and also in the columns denoted R, where the percentage of runs with $\lambda_f > 4$ is given. The results of the non-competitive standard DER, SADE, CRS8HC and SOMA are presented in Table 2.

Table 2. Standard differential evolution and other algorithms

Algorithm		DER				SADE				CRS8HC				SOMA			
Function	d	λ_f	λ_m	rne	R	λ_f	λ_m	rne	R	λ_f	λ_m	rne	R	λ_f	λ_m	rne	R
ackley	2	7.3	6.9	-2	100	7.3	6.9	-18	100	6.7	6.3	-49	100	2.7	2.5	386	27
dejong1	2	8.4	3.7	-1	100	8.5	3.8	-19	100	7.6	3.3	-50	100	6.7	3.1	707	80
griewangk	2	6.8	2.7	25	78	8.3	3.5	8	95	7.3	2.8	-51	95	5.9	2.4	1116	74
rastrig	2	8.5	4.9	-2	99	8.5	4.9	-18	99	7.5	4.4	-47	98	3.9	2.6	454	38
rosen	2	8.3	4.7	105	100	9.0	5.0	71	100	7.6	4.3	-51	100	3.4	2.1	1936	30
schwefel	2	7.5	5.5	-3	100	7.4	5.4	-18	98	7.4	5.4	-51	98	4.0	2.6	544	51
ackley	5	6.3	6.1	1	99	6.8	6.5	91	100	6.3	6.1	-12	100	3.4	3.0	1021	30
dejong1	5	7.1	3.2	-3	100	7.7	3.5	90	100	7.0	3.1	-13	100	9.5	4.8	1445	99
griewangk	5	5.2	1.7	14	70	7.8	2.9	127	100	5.0	1.7	-20	68	6.1	2.2	1057	73
rastrig	5	6.7	4.1	16	95	7.8	4.7	117	100	6.5	4.0	7	93	6.4	4.0	1099	72
rosen	5	7.2	4.4	528	100	8.1	4.9	511	100	7.2	4.3	-15	100	0.8	0.8	1506	0
schwefel	5	7.4	5.4	-3	98	7.5	5.5	91	100	7.4	5.4	-13	99	5.9	3.8	1275	81
ackley	10	5.9	5.7	14	99	6.5	6.3	248	100	6.3	6.1	0	100	4.3	4.0	1325	46
dejong1	10	6.5	3.0	6	100	7.2	3.3	252	100	6.9	3.2	4	100	10.7	6.2	1952	100
griewangk	10	5.3	1.6	18	78	7.2	2.4	260	100	6.6	2.1	-8	94	9.3	4.1	1429	95
rastrig	10	5.3	3.4	104	82	7.2	4.5	414	100	6.8	4.3	166	99	8.7	5.3	1509	93
rosen	10	6.7	4.3	429	100	7.4	4.7	729	100	6.8	4.3	-7	100	0.0	0.0	880	0
schwefel	10	7.3	5.2	9	96	7.5	5.5	264	100	7.5	5.5	30	100	6.8	4.6	1666	92
ackley	30	5.6	5.6	164	100	6.1	6.1	179	100	5.7	5.7	-68	95	1.6	1.4	324	0
dejong1	30	6.1	2.9	141	100	6.7	3.2	182	100	6.5	3.1	-66	100	5.8	2.7	667	100
griewangk	30	6.0	1.5	174	100	6.7	1.8	191	100	5.8	1.5	-67	87	3.6	0.3	485	19
rastrig	30	0.0	0.0	445	0	0.0	0.0	445	0	2.4	1.6	222	37	1.3	1.3	448	0
rosen	30	0.0	0.0	57	0	0.0	0.0	57	0	5.1	3.3	-20	98	0.0	0.0	58	0
schwefel	30	7.5	5.4	206	100	7.5	5.5	246	100	7.5	5.5	12	100	6.0	3.3	459	78

5 Discussion and Conclusions

As it is clear from Tables 1 and 2, three competitive variants of DE (DEBR18, DER9, and DEBEST9) are able to find the global minimum significantly more reliably than the other algorithms. They also outperformed the other algorithms regarding convergence rate (except DERADP3 and CRS8HC in several test tasks, but these algorithms exhibited less reliability). The performance of DEBR18, DER9 and DEBEST9 is apparently better than the performance of standard DER. DERADP3 was usually less reliable than the other competitive variants of differential evolution, very significantly in the case of Rosenbrock function.

The SADE algorithm was highly reliable in all the tasks at middle level of d ($d = 5$ and $d = 10$) but with significantly more time demand in comparison with DEBR18. In the case of Rastrigin and Rosenbrock function with $d = 30$, the SADE algorithm stopped due to the limit of maximal $ne = 600000$ without finding an acceptable approximation of the global minimum point like DER algorithm. Comparing with the standard differential evolution, SADE brings no improvement in tasks with $d = 30$.

As regards the comparison of DEBR18 and CRS8HC, CRS8HC searched for the global minimum with lower reliability at more time demand in the case of Rastrigin function (especially for $d = 30$) and also in several other tasks the reliability of CRS8HC was a bit lower, but with less time consumption.

SOMA performed worst in these test tasks. The reliability of SOMA was low except the easiest De Jong1 function, even at significantly higher time consumption comparing with the other algorithms under testing.

Among the three most reliable algorithms, the reliability of DEBR18 is the highest, but not significantly different from the reliability of DEBEST9 or DER9. Time demands of DERBR18 are less in all the test tasks when compared with DEBEST9. As regards comparison with DER9, DEBR18 worked significantly faster in two instances of the most time-consuming Rosenbrock function, comparable in the other two instances of this function, and only slightly slower in remaining tasks. DEBR18 can be recommended to next testing and perhaps to cautious application to practical tasks of the global optimization over continuous search space. The source code of DEBR18 in Matlab [9] is available at the web site `http://albert.osu.cz/tvrdik` and the source codes of DER9 or DEBEST9 can be easily derived from it.

The proposed competitive setting of the control parameters F and C proved to be an useful tool for self-adaptation of differential evolution, which can help to solve the global optimization tasks without necessity of fine parameter tuning. However, another research of self-adaptive differential evolution will proceed. The analysis of relative frequencies of different parameter settings (the frequencies are available in the experimental data but they have not yet been analyzed in detail) can bring knowledge on adaptive features of the competitive DE algorithm and it can also give some guide to propose such a subset of settings, which will outperform the variants of the algorithm described in this paper.

References

1. Ali M. M., Törn A.: *Population set based global optimization algorithms: Some modifications and numerical studies,* Computers and Operations Research **31** (2004) 1703 – 1725.
2. Bäck T.: *Evolutionary Algorithms in Theory and Practice,* Oxford University Press, New York (1996).
3. Brest J., Greimer S., Boškovič B., Mernik M., Žumer V.: *Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems,* IEEE Transactions on Evolutionary Computation (to appear).
4. Deb K.: *A population-based algorithm-generator for real parameter optimization,* Soft Computing **9** (2005) 236 – 253.
5. Lampinen J.: *A Bibliography of Differential Evolution Algorithm. Technical Report. Lappeenranta University of Technology, Department of Information Technology,* `http://www.lut.fi/~jlampine/debiblio.htm` (2002).
6. Liu J., Lampinen J.: *On Setting the Control Parameter of the Differential Evolution Method,* MENDEL 2002, 8th International Conference on Soft Computing (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2002) 11 – 18.
7. Liu J., Lampinen J.: *Adaptive Parameter Control of Differential Evolution,* MENDEL 2002, 8th International Conference on Soft Computing (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2002) 19 – 26.
8. Liu J., Lampinen J.: *A Fuzzy Adaptive Differential Evolution Algortithm,* Soft Computing **9** (2005) 448 – 462.
9. MATLAB, version 7 (R2006a), The MathWorks, Inc. (2006).
10. McCullough B. D., Wilson B.: *On the accuracy of statistical procedures in Microsoft Excel 2003,* Comput. Statist. and Data Anal. **49** (2005) 1244 – 1252.
11. Spall J. C.: *Introduction to Stochastic Search and Optimization,* Wiley-Intersience (2003).
12. Storn R., Price K.: *Differential evolution—a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,* J. Global Optimization **11** (1997) 341 – 359.
13. Šmuc T.: *Improving convergence properties of the differential evolution algorithm,* MENDEL 2002, 8th International Conference on Soft Computing (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2002) 80 – 86.
14. Tvrdík J., Mišík L., Křivý I.: *Competing Heuristics in Evolutionary Algorithms,* 2nd Euro-ISCI, Intelligent Technologies - Theory and Applications (Sinčák P. et al. eds.), IOS Press, Amsterdam (2002) 159 – 165.
15. Tvrdík J.: *Generalized controlled random search and competing heuristics. MENDEL 2004,* 10th International Conference on Soft Computing, (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2004) 228 – 233.
16. Tvrdík J.: *Competitive Differential Evolution,* MENDEL 2006, 12th International Conference on Soft Computing (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2006) 7 – 12.
17. Winter G., Galvan B., Alonso S., Gonzales B., Jimenez J. I., Greimer D.: *A flexible evolutionary agent: cooperation and competition among real-coded evolutionary operators,* Soft Computing **9** (2005) 299 – 323.

18. Wolpert D. H., Macready W. G.: *No Free Lunch Theorems for Optimization,* IEEE Transactions on Evolutionary Computation **1** (1997) 67 – 82.
19. Zaharie D.: *Critical Values for the Control Parameter of the Differential Evolution Algorithms,* MENDEL 2002, 8th International Conference on Soft Computing (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2002) 62 – 67.
20. Zaharie D.: *Control of popuplation diversity and adaptation in Differential Evolution Algorithms,* MENDEL 2003, 9th International Conference on Soft Computing (Matoušek R. and Ošmera P. eds). University of Technology, Brno (2003) 41 – 46.
21. Zelinka I., Lampinen J.: *SOMA – Self-Organizing Migrating Algorithm,* MENDEL 2000, 6th Int. Conference on Soft Computing, University of Technology, Brno (2000) 177 – 187.
22. Zelinka, I.: *Artificial intelligence in global optimization problems. (in Czech),* BEN, Praha (2002).