TASK QUARTERLY 11 No 1–2, 169–179

DIFFERENTIAL EVOLUTION WITH 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

(Received 21 December 2006; revised manuscript received 25 January 2007)

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

Keywords: global optimization, differential evolution, self-adaptation, numerical comparison

1. Introduction

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

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

There is point $\vec{x}^*$ to be found such that $\vec{x}^* = \arg\min_{\vec{x} \in S} f(\vec{x})$. Point $\vec{x}^*$ is called the global minimum point, while S is the search space. We focus on the problems wherein the objective function is continuous and the search space is a closed compact set, $S = \prod_{d=1}^{D} [a_d, b_d],\ a_d < b_d,\ d = 1, 2, \ldots, D$ (box constrains).

The problem of the global optimization is a difficult one and numerous stochastic algorithms have been proposed for its solution, see *e.g.* [1, 2]. The authors of many of these stochastic algorithms make claims about their efficiency and reliability in searching for the global minimum. Reliability means that the point with the minimal function's value found in the search process is sufficiently close to the global minimum point, while the measure of efficiency is the algorithm's finding a point sufficiently close to the global minimum point in reasonable time. However, when using such algorithms, one is faced with the problem of setting their control parameters. The efficiency and the reliability of many algorithms is strongly dependent on the values of their control parameters, but authors' recommendations in this respect are often

vague, see *e.g.* [3, 4]. The user is supposed to be able to adjust the parameter values according to the results of preliminary trial-and-error experiments with the search process. Such approach is not practicable in tasks, where global optimization is just one step on the way to solve the user's problem or when the user has no experience in the fine art of control parameter tuning.

Adaptive robust algorithms have been studied in recent years that are reliable enough at reasonable time-consumption without the necessity of fine tuning their input parameters. A proposal of an adaptive generator of robust algorithms was put forward in Deb [5]. Winter *et al.* [6] proposed a flexible evolutionary agent for real-coded genetic algorithms. Theoretical analysis of Wolpert and Macready [7] implies that no search algorithm can outperform others for all objective functions. In spite of this, there is empirical evidence that some algorithms can outperform others for relatively wide classes of problems both in the convergence rate and the reliability of finding the global minimum point. Thus, adaptive algorithms are likely to be found through experimental research rather than a purely theoretical approach.

The paper is a presentation of an adaptive procedure of setting control parameters in the algorithm of differential evolution in order to propose a self-adaptive algorithm for the box-constrained global optimization problem performing well for a wide range of problems without tuning control parameters. The following section is devoted to the algorithm of differential evolution and its control parameters. Competition of several control-parameter settings and its implementation into differential evolution are described in Section 3. The design of numerical experiments, the test functions used as benchmarks and the results are given in Section 4. In Section 5, summary results are presented and discussed for all the tested algorithms. Concluding remarks are made in the final section.

2. Differential Evolution and its control parameters

Differential Evolution (DE), as introduced by Storn and Price [3], is a simple but powerful evolutionary algorithm for global optimization over a box-constrained search space. The DE algorithm is written in pseudo-code as Algorithm 1. The basic idea of the differential evolution consists in alternating two populations, P and Q, of the same size, NP. A new trial point, $\vec{y}$, is composed of the current point, $\vec{x}_i$, of the old population and point $\vec{u}$ obtained by using mutation. If $f(\vec{y}) < f(\vec{x}_i)$, point $\vec{y}$ is inserted into the new population, Q, instead of $\vec{x}_i$. After completion of the new population, Q, the old population, P, is replaced by Q and the search continues until the stopping condition is fulfilled.

Algorithm 1. Differential evolution

```
1  generate P = (x_1, x_2, ..., x_NP); (NP points in S)
2  repeat
3      for i := 1 to NP 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 strategies to generate the mutant point $\vec{u}$. One of the most popular variants – called DE/rand/1/bin in the literature [8, 9, 3] and hereinafter denoted as DER – generates point $\vec{u}$ by adding the weighted difference of two points:

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

where $\vec{r}_1$, $\vec{r}_2$ and $\vec{r}_3$ are three distinct points taken randomly from P (not coincident with the current $\vec{x}_i$) and $F > 0$ is an input parameter. Another variant, called DE/best/2/bin and hereinafter denoted as DEBEST, generates point $\vec{u}$ according to the following formula:

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

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

In both the above mentioned strategies , the elements y_d, $d = 1, 2, \ldots, D$ of trial point $\vec{y}$ are built up by the crossover of its parents, $\vec{x}_i$ and $\vec{u}$, using the following rule:

$$y_d = \begin{cases} u_d & \text{if} \quad U_d \le CR \quad \text{or} \quad d = l \\ x_{id} & \text{if} \quad U_d > CR \quad \text{and} \quad d \ne l, \end{cases} \tag{3}$$

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 $CR \in [0,1]$ is an input parameter influencing the number of elements to be exchanged in the crossover. Equation (3) ensures that at least one element x_{id} of $\vec{x}_i$ is replaced with u_d, even if $CR = 0$.

Differential evolution has recently become one of the most popular algorithms for continuous global optimization problems [10, 9]. However, its search efficiency is known to be very sensitive to setting the F and CR values. The recommended values are $F = 0.8$ and $CR = 0.5$, but even Storn and Price used $0.5 \le F \le 1$ and $0 \le CR \le 1$ in their principal paper [3], depending on the results of preliminary tuning. They also set the population size below the recommended $NP = 10D$ in many of their test tasks.

Many papers have dealt with setting the control parameters for differential evolution. Ali and Törn [11] suggested to adapt the value of the scaling factor, F, in the search process according to the following 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, while $F_{\min}$ is an input parameter ensuring that $F \in [F_{\min}, 1]$. According to [11] this calculation of F reflects the demand to make the search more diversified at an early stage and more intensified at later stages, *i.e.* to produce larger values of F for large $f_{\max} - f_{\min}$, differences and smaller values of F otherwise. 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 a practical point of view, this occurs only as a brief episode

in the search process for most optimization tasks. Moreover, the values of F calculated using Equation (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 function is not changed by the shift. Nevertheless, the proposal (4) is an acceptable compromise between simplicity and performance.

Zaharie [12] derived the critical interval for DE control parameters, ensuring that the mean of population variance would not decrease, which resulted in the following relationship:

$$2pF^2 - \frac{2p}{NP} + \frac{p^2}{NP} > 0, \tag{5}$$

where $p = \max(1/D, CR)$ is the probability of "differential perturbation" according to Equation (3). Relationship (5) implies that the mean of population variance does not decrease if $F > \sqrt{1/NP}$, but practical reason for such result are very limited, as it brings no new information compared with the minimal value of $F = 0.5$ used in [3] and in other applications of differential evolution.

Other attempts at adaptating DE control parameters have appeared as well (see *e.g.* [13–16]). A recent state of adaptive parameter control in differential evolution has been summarized by Liu and Lampinen [17].

A new idea of self-adaptation of control parameters F and CR in differential evolution has been presented by Brest *et al.* [8]. The values of F and CR can be changed in each generation with respective probabilities τ_1, τ_2. New values of F are distributed uniformly in $[F_l, F_u]$ and new CR values are also uniform random values $\in [0, 1]$.

3. Competition in Differential Evolution

Control parameters can be set adaptively by implementing competition into the algorithm. This idea, similar to that of competition of local-search heuristics in evolutionary algorithms [18] or in controlled random search [19], has been proposed recently [20].

Let us have H settings (different values of F and CR used in the statements in line 4 and 5 of Algorithm 1) and choose among them at random with probability q_h, $h = 1, 2, \ldots, H$. The probabilities can be adjusted according to the success rate of setting in the preceding steps of the search process. The h^{th} setting is successful if it generates such a trial point $\vec{y}$ that $f(\vec{y}) < f(\vec{x}_i)$. When n_h is the current number of the h^{th} setting successes, 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 process's degeneration, the current values of q_h are reset to their starting values ($q_h = 1/H$) if any probability q_h decreases below a given limit $\delta > 0$.

It is supposed that such competition of various settings will prefer the successful ones. It provides a self-adaptive mechanism of setting control parameters at values appropriate for the problem being solved.

4. Experiments and results

Four variants of such competitive differential evolution were implemented and tested:

- DER9 – mutant vector $\vec{u}$ was generated according to Equation (1), nine settings of control parameters were all the combinations of three F values ($F = 0.5$, $F = 0.8$, and $F = 1$) with three values of CR,
- DEBEST9 – mutant vector $\vec{u}$ generated according to Equation (2), nine settings of control parameters F and CR as in DER9,
- DERADP3 – mutant vector $\vec{u}$ generated according to Equation (1), F adaptive according to Equation (4), three settings of CR,
- 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.

Three values of control parameter CR were used in all the variants, namely $CR = 0$, $CR = 0.5$, and $CR = 1$. The population's size was set at $NP = \max(20, 2D)$, and parameters for competition control were set to $n_0 = 2$, and $\delta = 1/(5H)$ in all the tasks.

The above 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 $CR = 0.5$, and the same population size as in the competitive variants of differential evolution. The second algorithm was self-adaptive differential evolution (SADE) as proposed by Brest *et al.*, with values of its control parameters recommended in [8], *i.e.* population size $NP = 10D$ and input parameters for self-adaptation set as follows: $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$, and $F_u = 0.9$. The third algorithm was that of controlled random search with eight competing local-search heuristics (CRS8HC), described in [19] including the setting of its control parameters. The last algorithm was SOMA (the all-to-one variant, see [21] and [4]). Its control parameters were set at $NP = 5D$, $mass = 3$, $step = 0.11$, and $prt = 0.1$, as recommended by Ivan Zelinka in a private communication.

The search for the global minimum was stopped if $f_{\max} - f_{\min} < 10^{-7}$ or the number of objective function evaluations exceeded the input's upper limit of $20\,000D$. The algorithms were tested on six functions commonly used as benchmarks (*cf.* [11, 9, 3]):

- Ackley's function (multimodal, separable)

$$f(\vec{x}) = -20\exp\left(-0.02\sqrt{\frac{1}{D}\sum_{d=1}^{D}x_d^2}\right) - \exp\left(\frac{1}{D}\sum_{d=1}^{D}\cos 2\pi x_d\right) + 20 + \exp(1)$$

$x_d \in [-30,30]$, $\vec{x}^* = (0,0,\ldots,0)$, $f(\vec{x}^*) = 0$,

- the First De Jong function (sphere model, unimodal, continuous, convex)

$$f(\vec{x}) = \sum_{d=1}^{D}x_d^2$$

$x_d \in [-5.12,5.12]$, $\vec{x}^* = (0,0,\ldots,0)$, $f(\vec{x}^*) = 0$,

– Griewank's function (multimodal, nonseparable)

$$f(\vec{x}) = \sum_{d=1}^{D} \frac{{x_d}^2}{4000} - \prod_{d=1}^{D} \cos\left(\frac{x_d}{\sqrt{d}}\right) + 1$$

$x_d \in [-400, 400]$, $\vec{x}^* = (0, 0, \ldots, 0)$, $f(\vec{x}^*) = 0$,

– Rastrigin's function (multimodal, separable)

$$f(\vec{x}) = 10\,D + \sum_{d=1}^{D} [{x_d}^2 - 10\cos(2\pi x_d)]$$

$x_d \in [-5.12, 5.12]$, $\vec{x}^* = (0, 0, \ldots, 0)$, $f(\vec{x}^*) = 0$,

– Rosenbrock's function (banana valley, unimodal, nonseparable)

$$f(\vec{x}) = \sum_{d=1}^{D-1} \left[100(x_d^2 - x_{d+1})^2 + (1 - x_d)^2\right]$$

$x_d \in [-2048, 2048]$, $\vec{x}^* = (1, 1, \ldots, 1)$, $f(\vec{x}^*) = 0$ and

– Schwefel's function (multimodal, the global minimum distant from the next best local minima)

$$f(\vec{x}) = \sum_{d=1}^{D} x_d \sin(\sqrt{|\,x_d\,|})$$

$x_d \in [-500, 500]$, $\vec{x}^* = (s, s, \ldots, s)$, $s = 420.9687$, $f(\vec{x}^*) = -418.9829\,D$.

The tests were preformed for all the functions at four levels of dimension D of the search spaces, namely $D = 2$, $D = 5$, $D = 10$ and $D = 30$. A hundred independent runs were carried out for each function and level of D.

The accuracy of result obtained in the search for the global minimum was evaluated according to the number of duplicated digits compared with the correct result. The number of duplicated digits, λ, can be calculated via *log relative error* [22]:

– if $c \neq 0$, 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 correct value and m denotes the value obtained by the search;

– if $c = 0$, 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 given in the results: λ_f for the function value and λ_m, being the minimal λ for the global minimum point $(x_1, x_2, \ldots, x_D)$ found by the search.

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
griewank	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
griewank	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
griewank	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
griewank	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 results of competitive DE are presented in Table 1. Time consumption is expressed as the average number, ne, of the objective function's evaluations required to attain the stopping condition. In order to facilitate a comparison of the algorithms, the ne values are given only for the DEBR18 algorithm, while the percentage change of ne relative to DEBR18 is presented for the other algorithms in the tables columns marked rne. Accordingly, negative values of rne mean less and positive values mean more time consumed compared with DEBR18. For example, the value of $rne = -50$ means half the ne, while the value of $rne = 100$ means that ne is twice that in DEBR18. The number of objective function evaluations can be easily recalculated as $ne = ne_0 \times (1 + rne/100)$, where ne_0 is the appropriate number of objective function evaluations for DEBR18 in Table 1.

The search's reliability is given in the columns of λ_f, λ_m and R, the average values of λ_f and λ_m in a hundred runs. R is the percentage of runs with $\lambda_f > 4$. The results for the other algorithms, *i.e.* the non-competitive standard DER, SADE, CRS8HC and SOMA, are presented in Table 2.

5. Discussion

It follows from Tables 1 and 2 and the summary of Table 3 that three competitive variants of DE (DEBR18, DER9, and DEBEST9) are capable of finding the global minimum significantly more reliably than the other algorithms. They have also outperformed the other algorithms in the convergence rate (except for DERADP3 and CRS8HC in several test tasks, but these algorithms are less reliable). The performance

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
griewank	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
griewank	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
griewank	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
griewank	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

of DEBR18, DER9 and DEBEST9 has been apparently better than that of the standard DER. DERADP3 has usually been less reliable than the other competitive variants of differential evolution, quite significantly in the case of Rosenbrock's function (see Table 1).

The SADE algorithm was highly reliable in all the tasks at the middle level of D ($D = 5$ and $D = 10$), but has had significantly higher time requirements than DEBR18. In the case of Rastrigin's and Rosenbrock's functions with $D = 30$, the SADE algorithm stopped due to the limit of maximal $ne = 600\,000$ without finding an acceptable approximation of the global minimum point. The same happened in the case of the DER algorithm. When we compare SADE and the standard differential evolution (DER), the former offered no improvement in the tasks with $D = 30$.

Compared with DEBR18, CRS8HC searched for the global minimum with lower reliability and higher time requirements in the case of Rastrigin's function (especially for $D = 30$). In most other tasks its reliability was the same or slightly lower, but mostly with less time consumed.

SOMA performed the worst of all the algorithms in these test tasks. Its reliability was low except for the easiest De Jong1 function, even at time consumption significantly higher than that of the other tested algorithms.

A summary comparison of the algorithms is shown in Figure 1. The required property of the algorithm has been to find a good approximation of the global minimum with high reliability and low time consumption. From this point of view, three competitive variants of DE (DEBR18, DER9, and DEBEST9) have outperformed

Table 3. Summary of results – averages of λ_f, λ_m, *rne* and R for all the test tasks

Algorithm	λ_f	λ_m	*rne*	R
DEBR18	7.1	4.3	0	99.8
DER9	7.0	4.3	−4	99.4
DEBEST9	7.1	4.4	19	99.8
DERADP3	6.2	3.8	5	86.2
DER	6.2	3.8	102	87.3
SADE	6.9	4.2	187	91.3
CRS8HC	6.6	4.1	−7	94.2
SOMA	4.9	2.8	990	53.3

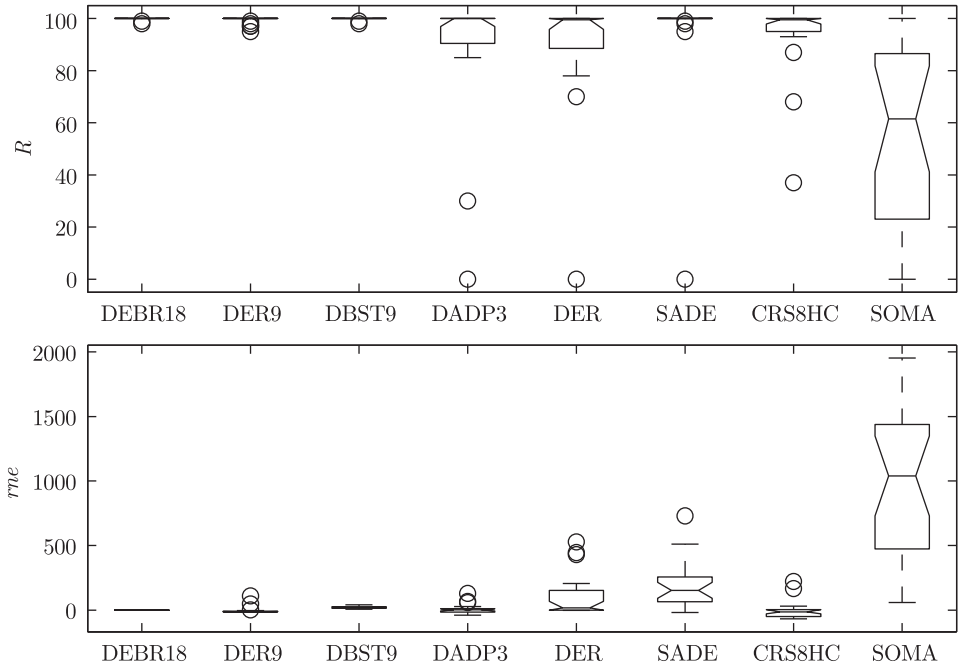

Figure 1. Comparison of algorithms – box plots of R and *rne*

the other algorithms significantly in that either their reliability is higher or the number of objective function evaluations required to reach the stopping condition is lower. They also have much less variability of R and *rne*, an indication of their robustness.

6. Conclusions

Among the three most reliable algorithms, the reliability of DEBR18 is the highest, though not significantly different from the reliability of DEBEST9 or DER9. The time required by DERBR18 has been less than that required by DEBEST9 in all the test tasks. Compared with DER9, DEBR18 worked significantly faster in two instances of the most time-consuming Rosenbrock function, was comparable in the other two instances of this function and only slightly slower in the remaining tasks. DEBR18 and DER9 can be recommended for cautious application in practical tasks of global optimization over continuous search spaces. The source codes of DEBR18 and DER9 in Matlab [23] are included in the program library of self-adaptive

stochastic algorithms (Stochastic Algorithms for Global Optimization – MATLAB Library, http://albert.osu.cz/oukip/optimization), freely accessible subject to the conditions of the GNU license.

The proposed competitive setting of control parameters F and CR has proved to be a useful tool for self-adaptation of differential evolution, helpful in solving global optimization tasks without the necessity for fine control-parameter tuning. However, further research of self-adaptive differential evolution will proceed. Further analysis of relative frequencies of various parameter settings may shed more light on adaptive features of the competitive DE algorithm and it be instrumental in finding a subset of settings capable of outperforming the algorithm's variants described in this paper.

Acknowledgements

This research 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.

References

[1] Bäck T 1996 *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York
[2] Spall J C 2003 *Introduction to Stochastic Search and Optimization*, Wiley-Intersience
[3] Storn R and Price K 1997 *J. Global Optimization* **11** 341
[4] Zelinka I 2002 *Artificial Intelligence in Global Optimization Problems*, BEN, Praha (in Czech)
[5] Deb K 2005 *Soft Computing* **9** 236
[6] Winter G, Galvan B, Alonso S, Gonzales B, Jimenez J I and Greimer D 2005 *Soft Computing* **9** 299
[7] Wolpert D H and Macready W G 1997 *IEEE Trans. on Evolutionary Computation* **1** 67
[8] Brest J, Greimer S, Boškovič B and Mernik M 2006 *IEEE Trans. on Evolutionary Computation* **10** (6) 646
[9] Price K V, Storn R and Lampinen J 2005 *Differential Evolution: A Practical Approach to Global Optimization*, Springer-Verlag
[10] Lampinen J 2002 *A Bibliography of Differential Evolution Algorithm. Technical Report*, Lappeenranta University of Technology, Department of Information Technology, http://www.lut.fi/∼jlampine/debiblio.htm
[11] Ali M M and Törn A 2004 *Comput. Operations Res.* **31** 1703
[12] Zaharie D 2002 *MENDEL 2002, 8th Int. Conf. on Soft Computing*, (Matoušek R and Ošmera P, Eds), University of Technology, Brno, pp. 62–67
[13] Liu J and Lampinen J 2002 *MENDEL 2002, 8th Int. Conf. on Soft Computing*, (Matoušek R and Ošmera P, Eds), University of Technology, Brno, pp. 11–18
[14] Liu J and Lampinen J 2002 *MENDEL 2002, 8th Int. Conf. on Soft Computing*, (Matoušek R and Ošmera P, Eds), University of Technology, Brno, pp. 19–26
[15] Šmuc T 2002 *MENDEL 2002, 8th Int. Conf. on Soft Computing*, (Matoušek R and Ošmera P, Eds), University of Technology, Brno, pp. 80–86
[16] Zaharie D 2003 *MENDEL 2003, 9th Int. Conf. on Soft Computing*, (Matoušek R and Ošmera P, Eds), University of Technology, Brno, pp. 41–46
[17] Liu J and Lampinen J 2005 *Soft Computing* **9** 448
[18] Tvrdík J, Mišík L and Křivý I 2002 *2nd Euro-ISCI, Intelligent Technologies – Theory and Applications*, (Sinčák P *et al.*, Eds), IOS Press, Amsterdam, pp. 159–165
[19] Tvrdík J 2004 *MENDEL 2004, 10th Int. Conf. on Soft Computing*, (Matoušek R and Ošmera P, Eds), University of Technology, Brno, pp. 228–233
[20] Tvrdík J 2006 *MENDEL 2006, 12th Int. Conf. on Soft Computing*, (Matoušek R and Ošmera P, Eds), University of Technology, Brno, pp. 7–12

[21] Zelinka I and Lampinen J 2000 *MENDEL 2000, 6th Int. Conf. on Soft Computing*, University of Technology, Brno, pp. 177–187
[22] McCullough B D and Wilson B 2005 *Comput. Statist. and Data Anal.* **49** 1244
[23] 2006 *MATLAB, version 7 (R2006a)*, The MathWorks, Inc.