COMPETITIVE DIFFERENTIAL EVOLUTION

Josef Tvrdík

University of Ostrava, Department of Computer Science
30. dubna 22, 701 03 Ostrava, Czech Republic
phone: +420/596160231, fax: +420/596120478
e-mail: tvrdik@osu.cz

Abstract. This paper deals with differential evolution. Adaptation of its controlling parameters was studied. The competition of different controlling-parameter settings was proposed and tested on six functions at three levels of the search space dimensions. The competitive differential evolution proved to be more reliable and less time-consuming than the standard differential evolution. The competitive differential evolution also outperformed some other algorithms that were tested.

Keywords: global optimization, differential evolution, competition, adaptive algorithms.

Introduction

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

\[ f : D \rightarrow \mathbb{R}, \quad D \subset \mathbb{R}^d \]

the point \( x^* \) is to be found such that \( x^* = \arg \min_{x \in D} f(x) \). The point \( x^* \) is called the global minimum point and \( D \) is the search space. We focus on the problems, where \( D \) is closed compact set defined as

\[ D = \prod_{i=1}^{d} [a_i, b_i], \quad a_i < b_i, \quad i = 1, 2, \ldots, d, \]

the objective function is continuous and function value \( f(x) \) can be evaluated at any point \( x \in D \).

The authors of many stochastic search algorithms for solving this problem 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. However, when we use such algorithms, we face the problem of the setting their controlling parameters. The efficiency and the reliability of many algorithms is strongly dependent on the values of controlling parameters. Recommendations given by authors are often vague or uncertain, see e.g. [9], [18]. A user is supposed to be able to change the parameter values according to partial results obtained in 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 [2]. Winter et al. [13] 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 [14]. 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. Thus, the way to the adaptive algorithms leads rather trough the experimental research than purely theoretical approach.

Differential Evolution and its Controlling 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 \( y \) is composed of the current point \( x_i \) of old population and the point \( u \) obtained by using mutation. If \( f(y) < f(x_i) \) the point \( y \) is inserted into the new population \( Q \) instead of \( x_i \). After completion of the new population \( Q \) the old population \( P \) is replaced by \( Q \) and the search continues until stopping condition is fulfilled.
Algorithm 1. Differential evolution

1. generate $P = (x_1, x_2, \ldots, x_N)$; ($N$ points in $D$ at random)
2. repeat
3. for $i := 1$ to $N$ do
4.  compute a mutant vector $u$;
5.  create a trial point $y$ 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;

The most popular variant of DE (called DER in this text) generates the point $u$ by adding the weighted difference of two points

$$u = r_1 + F(r_2 - r_3),$$

where $r_1, r_2$ and $r_3$ are three distinct points taken randomly from $P$ (not coinciding with the current $x_i$) and $F > 0$ is an input parameter. Another variant called DEBEST generates the point $u$ by the following rule

$$u = x_{\min} + F(r_1 + r_2 - r_3 - r_4),$$

where $r_1, r_2, r_3, r_4$ are four distinct points taken randomly from $P$ (not coinciding with the current $x_i$), $x_{\min}$ 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 $y$ are built up by the crossover of its parents $x_i$ and $u$ using the following rule

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

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.

The differential evolution has become one of the most popular algorithms for the continuous global optimization problems in recent years, see [3]. 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 [9] use $0.5 \leq F \leq 1$ and $0 \leq C \leq 1$. They also set the size of population less than the recommended $N = 10d$ 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 search process according to the equation

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

where $f_{\min}, f_{\max}$ are respectively the minimum and maximum function values in the population and $F_{\min}$ is an input parameter ensuring $F \in [F_{\min}, 1)$. This adaptive attempt is useful only sometimes and there are tasks where such adaptation is not helpful, see [12].

Zaharie [15] derived the critical interval for the controlling parameters of DE. This interval ensures to keep the mean of population variance non-decreasing, which results in the following relationship

$$2pF^2 - \frac{2p}{N} + \frac{p^2}{N} > 0,$$

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 [9] and in other applications of differential evolution.

Some attempts to the adaptation of DE control parameters have appeared on MENDEL recently, see [4], [5], [8], and [16]. Up-to-date state of adaptive parameter control in differential evolution is summarized by Liu and Lampinen [6].
Competition in Differential Evolution

The setting of the controlling parameters can be made adaptive through the implementation of a competition into the algorithm. This idea is similar to the competition of local-search heuristics in evolutionary algorithm [10] or in controlled random search [11].

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 \( q_i \), \( i = 1, 2, \ldots, h \). The probabilities can be changed according to the success of the setting in preceding steps of search process. The \( i \)-th setting is successful if it generates such a trial point \( y \) that \( f(y) < f(x_i) \). When \( n_i \) is the current number of the \( i \)-th setting successes, the probability \( q_i \) can be evaluated simply as the relative frequency

\[
q_i = \frac{n_i + n_0}{\sum_{j=1}^{h}(n_j + n_0)},
\]

where \( n_0 > 0 \) is a constant. The setting of \( n_0 \geq 1 \) prevents a dramatic change in \( q_i \) by one random successful use of the \( i \)-th parameter setting. In order to avoid the degeneration of process the current values of \( q_i \) are reset to their starting values (\( q_i = 1/h \)) if any probability \( q_i \) decreases below 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 adaptive mechanism of setting control parameter appropriate to the problem actually solved.

Experiments and Results

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

- \( \text{DER9} \) – the mutant vector \( u \) is generated according to (1), nine settings of controlling 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,
- \( \text{DEBEST9} \) – the mutant vector \( u \) is generated according to (2), nine settings of controlling parameters \( F \) and \( C \) like in \( \text{DER9} \),
- \( \text{DERADP3} \) – the mutant vector \( u \) is generated according to (1), \( F \) adaptive according to (4), three settings of \( C \).
- \( \text{DEBR18} \) – 18 settings, aggregation of settings used in \( \text{DER9} \) and \( \text{DEBEST9} \), implemented due to the good performance of \( \text{DER9} \) and \( \text{DEBEST9} \) in the test tasks.

Population size was set to \( N = \max(20, 2d) \), and parameters for competition control were set to \( n_0 = 2 \), and \( \delta = 1/45 \). These variants of competitive differential evolution were compared with three other algorithms. One of them was the standard \( \text{DER} \) 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 the \( \text{CRS8HC} \) with eight competing local-search heuristics, described in [11] including the setting of its controlled parameters.

The last algorithm was \( \text{SOMA} \) (all-to-one variant), see [17] and [18], with its controlling parameters set to recommended values (\( N = 5d, \text{mass}=1.9, \text{step}=0.3, \text{prt}=0.5 \)). The search for the global minimum was stopped if \( f_{\text{max}} - f_{\text{min}} < 1E-07 \) or the number of objective function evaluations exceeds the input upper limit \( 200000d \).

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 [9] or [1]. The search spaces \( D \) in all test tasks were symmetric and with the same interval in each dimension, \( a_{ij} = -b_i, b_i = b_j, i, j = 1, 2, \ldots, d \). The values of \( b_i \) were set as follows: 2.048 for Rosenbrock’s function, 5.12 for first De Jong and Rastrigin function, 30 for Ackley function, 400 for Griewank function and 500 for Schweefel function. The test were preformed for all the functions at three level of dimension \( d \) of search spaces, namely \( d = 5, d = 10 \) and \( d = 30 \). One hundred of repeated runs were carried out for each function and level of \( d \).

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 \( \lambda \) can be calculated via \( \log \text{relative error} \) [7]:

\[
\lambda = -\log_{10} \frac{\text{relative error}}{\text{certified result}}.
\]
• If $c \neq 0$, the $\lambda$ is evaluated as

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

(7)

where $c$ denotes the certified value and $m$ denotes the value obtained by the search.

• If $c = 0$, the $\lambda$ is evaluated as

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

(8)

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

Table 1: Competitive differential evolution algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
<th>$d$</th>
<th>DER9 $\lambda_f$</th>
<th>DER9 $\lambda_m$</th>
<th>DEBEST9 $\lambda_f$</th>
<th>DEBEST9 $\lambda_m$</th>
<th>DERADP3 $\lambda_f$</th>
<th>DERADP3 $\lambda_m$</th>
<th>$rne$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ackley</td>
<td>-13</td>
<td>100</td>
<td>6.0</td>
<td>5.9</td>
<td>21</td>
<td>100</td>
<td>6.1</td>
<td>6.0</td>
<td>-35</td>
<td>100</td>
</tr>
<tr>
<td>dejong1</td>
<td>-13</td>
<td>100</td>
<td>6.5</td>
<td>3.1</td>
<td>21</td>
<td>100</td>
<td>6.7</td>
<td>3.1</td>
<td>-12</td>
<td>100</td>
</tr>
<tr>
<td>griewangk</td>
<td>-13</td>
<td>100</td>
<td>6.5</td>
<td>1.7</td>
<td>24</td>
<td>100</td>
<td>6.7</td>
<td>1.8</td>
<td>-12</td>
<td>100</td>
</tr>
<tr>
<td>rastrig</td>
<td>-12</td>
<td>100</td>
<td>6.5</td>
<td>4.2</td>
<td>25</td>
<td>100</td>
<td>6.7</td>
<td>4.3</td>
<td>-14</td>
<td>100</td>
</tr>
<tr>
<td>rosen</td>
<td>1</td>
<td>100</td>
<td>6.4</td>
<td>4.3</td>
<td>28</td>
<td>100</td>
<td>0.0</td>
<td>0.0</td>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>schwefel</td>
<td>-12</td>
<td>100</td>
<td>7.5</td>
<td>5.5</td>
<td>20</td>
<td>100</td>
<td>7.5</td>
<td>5.5</td>
<td>-38</td>
<td>100</td>
</tr>
<tr>
<td>ackley</td>
<td>-15</td>
<td>100</td>
<td>6.1</td>
<td>5.9</td>
<td>24</td>
<td>100</td>
<td>5.6</td>
<td>5.4</td>
<td>-25</td>
<td>89</td>
</tr>
<tr>
<td>dejong1</td>
<td>-14</td>
<td>100</td>
<td>6.7</td>
<td>3.1</td>
<td>22</td>
<td>100</td>
<td>7.1</td>
<td>3.2</td>
<td>-3</td>
<td>100</td>
</tr>
<tr>
<td>griewangk</td>
<td>-18</td>
<td>100</td>
<td>6.8</td>
<td>2.2</td>
<td>37</td>
<td>100</td>
<td>7.0</td>
<td>2.4</td>
<td>-5</td>
<td>93</td>
</tr>
<tr>
<td>rastrig</td>
<td>-13</td>
<td>100</td>
<td>6.6</td>
<td>4.2</td>
<td>25</td>
<td>99</td>
<td>6.8</td>
<td>4.2</td>
<td>-5</td>
<td>96</td>
</tr>
<tr>
<td>rosen</td>
<td>110</td>
<td>95</td>
<td>6.4</td>
<td>4.2</td>
<td>15</td>
<td>100</td>
<td>0.1</td>
<td>0.1</td>
<td>130</td>
<td>2</td>
</tr>
<tr>
<td>schwefel</td>
<td>-14</td>
<td>97</td>
<td>7.4</td>
<td>5.4</td>
<td>21</td>
<td>98</td>
<td>6.8</td>
<td>4.9</td>
<td>-33</td>
<td>89</td>
</tr>
<tr>
<td>ackley</td>
<td>-11</td>
<td>100</td>
<td>6.5</td>
<td>6.2</td>
<td>17</td>
<td>100</td>
<td>6.6</td>
<td>6.3</td>
<td>-7</td>
<td>100</td>
</tr>
<tr>
<td>dejong1</td>
<td>-12</td>
<td>100</td>
<td>7.2</td>
<td>3.2</td>
<td>14</td>
<td>100</td>
<td>7.4</td>
<td>3.3</td>
<td>13</td>
<td>100</td>
</tr>
<tr>
<td>griewangk</td>
<td>-15</td>
<td>99</td>
<td>7.2</td>
<td>2.6</td>
<td>40</td>
<td>100</td>
<td>7.2</td>
<td>2.7</td>
<td>10</td>
<td>93</td>
</tr>
<tr>
<td>rastrig</td>
<td>-13</td>
<td>100</td>
<td>7.2</td>
<td>4.4</td>
<td>18</td>
<td>100</td>
<td>7.4</td>
<td>4.5</td>
<td>1</td>
<td>99</td>
</tr>
<tr>
<td>rosen</td>
<td>47</td>
<td>97</td>
<td>6.8</td>
<td>4.2</td>
<td>14</td>
<td>99</td>
<td>3.1</td>
<td>2.0</td>
<td>139</td>
<td>44</td>
</tr>
<tr>
<td>schwefel</td>
<td>-12</td>
<td>98</td>
<td>7.4</td>
<td>5.4</td>
<td>12</td>
<td>99</td>
<td>7.4</td>
<td>5.4</td>
<td>-28</td>
<td>98</td>
</tr>
</tbody>
</table>

Table 2: Competitive differential evolution – DEBR18

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Function</th>
<th>$d = 30$</th>
<th>$d = 10$</th>
<th>$d = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\lambda_f$</td>
<td>$\lambda_m$</td>
<td>$rne$</td>
<td>$R$</td>
</tr>
<tr>
<td>ackley</td>
<td>5.9</td>
<td>5.8</td>
<td>142208</td>
<td>100</td>
</tr>
<tr>
<td>dejong1</td>
<td>6.4</td>
<td>3.0</td>
<td>78664</td>
<td>100</td>
</tr>
<tr>
<td>griewangk</td>
<td>6.4</td>
<td>1.6</td>
<td>103095</td>
<td>100</td>
</tr>
<tr>
<td>rastrig</td>
<td>6.4</td>
<td>4.1</td>
<td>110071</td>
<td>100</td>
</tr>
<tr>
<td>rosen</td>
<td>6.3</td>
<td>4.3</td>
<td>381972</td>
<td>100</td>
</tr>
<tr>
<td>schwefel</td>
<td>7.5</td>
<td>5.4</td>
<td>108050</td>
<td>100</td>
</tr>
</tbody>
</table>

The results of the competitive DE are presented in Table 1 and 2. The time consumption is expressed as the average number ($rne$) of the objective function evaluations needed to reach the stopping condition. Columns of Table 1 (and also Table 3) denoted $rne$ contain the relative change of $rne$ in percents when compared with DEBR18 in Table 2. Therefore, the negative values of $rne$ mean less time consumption with respect
Table 3: Standard differential evolution, CRS and SOMA

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DER</th>
<th>CRS8HC</th>
<th>SOMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(d)</td>
<td>(\lambda_f)</td>
<td>(\lambda_m)</td>
</tr>
<tr>
<td>ackley</td>
<td>30</td>
<td>5.6</td>
<td>5.6</td>
</tr>
<tr>
<td>dejong1</td>
<td>30</td>
<td>6.1</td>
<td>2.9</td>
</tr>
<tr>
<td>griewangk</td>
<td>30</td>
<td>6.0</td>
<td>1.5</td>
</tr>
<tr>
<td>rastrig</td>
<td>30</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>rosen</td>
<td>30</td>
<td>0.0</td>
<td>57</td>
</tr>
<tr>
<td>schwefel</td>
<td>30</td>
<td>7.5</td>
<td>5.4</td>
</tr>
<tr>
<td>ackley</td>
<td>10</td>
<td>5.9</td>
<td>5.7</td>
</tr>
<tr>
<td>dejong1</td>
<td>10</td>
<td>6.5</td>
<td>3.0</td>
</tr>
<tr>
<td>griewangk</td>
<td>10</td>
<td>5.3</td>
<td>1.6</td>
</tr>
<tr>
<td>rastrig</td>
<td>10</td>
<td>5.3</td>
<td>3.4</td>
</tr>
<tr>
<td>rosen</td>
<td>10</td>
<td>6.7</td>
<td>4.3</td>
</tr>
<tr>
<td>schwefel</td>
<td>10</td>
<td>7.3</td>
<td>5.2</td>
</tr>
<tr>
<td>ackley</td>
<td>5</td>
<td>6.3</td>
<td>6.1</td>
</tr>
<tr>
<td>dejong1</td>
<td>5</td>
<td>7.1</td>
<td>3.2</td>
</tr>
<tr>
<td>griewangk</td>
<td>5</td>
<td>5.2</td>
<td>1.7</td>
</tr>
<tr>
<td>rastrig</td>
<td>5</td>
<td>6.7</td>
<td>4.1</td>
</tr>
<tr>
<td>rosen</td>
<td>5</td>
<td>7.2</td>
<td>4.4</td>
</tr>
<tr>
<td>schwefel</td>
<td>5</td>
<td>7.4</td>
<td>5.4</td>
</tr>
</tbody>
</table>

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 \(\lambda_f\) and \(\lambda_m\), 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, CRS8HC and SOMA are presented in Table 3. The columns denoted \(rne\) also contain the relative change of \(ne\) in percents when compared with DEBR18.

Discussion and Conclusions

As it is seen in Tables 1, 2 and 3, three competitive variants of DE (DEBR18, DER9, and DEBEST9) searched for the global minimum significantly more reliable than the other algorithms. They also outperformed the other algorithms in convergence rate (except DERADP3 and CRS8HC in some test tasks, but these algorithms exhibited less reliability in some tasks). 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.

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 badly in these test tasks, although its controlling parameters were set to the middle of recommended intervals [18]. The reliability of SOMA was very poor except the easiest De Jong1 function, even at more time consumption comparing with the other algorithms under testing.

The winner among the algorithms is DEBR18 algorithm. Its reliability is the highest (but not significantly different from the reliability of DEBEST9 or DER9). Its time demands are less on all the test tasks when compared with DEBEST9. As regards comparison with DER9, DEBR18 worked faster in the case of the most time-consuming Rosenbrock function and only slightly slower in remaining tasks. The source code of DEBR18 in Matlab is presented in the full CD version of this contribution.

The proposed competitive setting of the controlling parameters \(F\) and \(C\) has proved to be useful idea, which can help to solve the global optimization tasks without necessity of fine parameter tuning. However, another research of competitive differential evolution should continue.
References


Acknowledgment

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.