Rotation-Based Learning: A Novel Extension of Opposition-Based Learning

Huichao Liu[1,2,3], Zhijian Wu[1,2], Huanzhe Li[1,2], Hui Wang[1,2,4], Shahryar Rahnamayan[5], and Changshou Deng[6]

[1] Computer School of Wuhan University, Wuhan 430072, China
[2] State Key Lab. of Software Engineering, Wuhan University, Wuhan, China
{huichaoliu,zhijianwu}@whu.edu.cn
[3] College of Information Engineering, Huanghuai University, Zhumadian, China
[4] School of Inform. Engin., Nanchang Institute of Technology, Nanchang, China
[5] Department of Electrical, Computer and Software Engineering, University of Ontario Institute of Technology, Oshawa, Canada
[6] School of Information Science and Technology, Jiujiang University, Jiujiang, China

Abstract. Opposition-based learning (OBL) scheme is an effective mechanism to enhance soft computing techniques, but it also has some limitations. To extend the OBL scheme, this paper proposes a novel rotation-based learning (RBL) mechanism, in which a rotation number is achieved by applying a specified rotation angle to the original number along a specific circle in two-dimensional space. By assigning different angles, RBL can search any point in the search space. Therefore, RBL could be more flexible than OBL to find the promising candidate solutions in the complex search spaces. In order to verify its effectiveness, the RBL mechanism is embedded into differential evolution (DE) and the rotation-based differential evolution (RDE) algorithm is introduced. Experimental studies are conducted on a set of widely used benchmark functions. Simulation results demonstrate the effectiveness of RBL mechanism, and the proposed RDE algorithm performs significantly better than, or at least comparable to, several state-of-the-art DE variants.

Keywords: Evolutionary computation, differential evolution, rotation-based learning, opposition-based learning.

1 Introduction

In the past decades, some researchers have proposed many excellent nature-inspired algorithms which have shown better performance compared to traditional methods when tackling the complex problems. But, there is not any algorithm can solve all kinds of optimization problems with the same efficiency level due to their own disadvantages. In order to further performance improvement and expanding the application scopes of classical algorithms, a variety of enhanced mechanisms have been proposed. The opposition-based learning (OBL) is an effective enhancement scheme for optimization process, and it was introduced by Tizhoosh [10]. OBL has been used to various fields, such as DE [7],

D.-N. Pham and S.-B. Park (Eds.): PRICAI 2014, LNAI 8862, pp. 511–522, 2014.

PSO [17], ABC [15], ACO [3], etc. However, the OBL mechanism also has its limitations that it can only search one opposite point in the opposition space. So that it could be inefficient in some situations, e.g. if the domain and search space of a problem are symmetrical, then putting the OBL into effect is meaningless.

In this paper, a novel rotation-based learning mechanism, called RBL, is proposed by extending the classical OBL mechanism. Using RBL mechanism, a rotation number can be obtained by rotating anticlockwise the original number by the specified angle on a specific circle in two-dimensional space. Hence, RBL can find any point in the search space by rotating different angles. Moreover, RBL also has several different application modes, e.g. OBL can be seen as a special case of RBL when the rotation angle is equal to 180 degrees.

An important goal in this paper is to verify the effectiveness of RBL mechanism. To this end, we embed the RBL mechanism into DE, and introduce the rotation-based differential evolution (RDE) algorithm. Experimental studies are conducted on a suite of 13 global optimization problems. The experiment results show that the RDE algorithm outperforms the other algorithms on the majority of the test problems.

2 Related Work

2.1 Differential Evolution

As a population-based search method, DE firstly produces an initial vector population, in which the values of each dimension for all individuals are randomly sampled within the search space. Then, DE evolves the subsequent populations until the stop criteria are met. Assume that X_i^G (i=1,2,...,NP) is the ith individual in population $P(G)$ (NP is the population size, and G is the generation index). The *DE/rand/1/exp* scheme used in this paper is described as follows.

Mutation– It is used to produce the mutant (or donor) vector V_i^G as

$$V_i^G = X_{r_1}^G + F \cdot (X_{r_2}^G - X_{r_3}^G), \tag{1}$$

where r_1, r_2 and r_3 are randomly selected from [1, NP], and they plus i are mutually different. F is amplification factor, and $(X_{r_1}^G - X_{r_2}^G)$ is differential vector.

Crossover–DE utilizes the crossover operation to generate new trial vector $U_i^G = (U_{1i}^G, U_{2i}^G, \ldots, U_{Di}^G)$ (D indicates problem dimension). In which, each component is defined by

$$U_{ji}^G = \begin{cases} V_{ji}^G \text{ , if } j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D \\ X_{ji}^G \text{ , otherwise} \end{cases} \tag{2}$$

where n acts as the starting point of exchange in target vector, and L denotes the number of components that donor vector actually contributes to target vector.

Selection–It is an approach which decides which vector (U_i^G or X_i^G) should be a member of next generation G+1. For the minimization problems, the selection operator is defined by the following greedy mechanism:

$$X_i^{G+1} = \begin{cases} U_i^G \text{ , if } f(U_i^G) \leq f(X_i^G) \\ X_i^G \text{ , otherwise} \end{cases} \tag{3}$$

Ever since differential evolution (DE) was proposed by Price and Storn [9] in 1995, vast DE variants have been introduced to optimize the various benchmark and real-world problems. A comprehensive explanation of DE and its variants can be found in the survey [2].

2.2 Opposition-Based Learning (OBL)

The main idea behind OBL is the simultaneous consideration of an estimation and its opposite estimation in order to provide another chance to create a candidate solution closer to the global optimum. In [8], Rahnamayan et al. proposed an Euclidean distance-to-optimal solution proof that shows intuitively why considering the opposite of a candidate solution is more beneficial than another random solution. In OBL mechanism, the key concepts are described as follows.

Opposite Number: Let $z \in [a, b]$ be a real number. The opposite number $\overline{z}$ is defined by

$$\overline{z} = a + b - z. \tag{4}$$

Obviously, the definition of the opposite number can be extended to higher dimensions.

Opposite Point: Let $Z = (z_1, z_2, \ldots, z_D)$ be a point in the D-dimensional search space, where $z_j \in R_j = [a_j, b_j]$, $j \in [1, 2, \ldots, D]$ and $Z \in S = \prod_{j=1}^{D} R_j$. The opposite point $\overline{Z}= (\overline{z}_1, \overline{z}_2, \ldots, \overline{z}_D)$ is defined by

$$\overline{z}_j = a_j + b_j - z_j. \tag{5}$$

Note that the OBL mechanism is not used in every generation during the search process. In some opposition-based optimization algorithms, OBL operation is often conducted by the specified probability parameter, known as *jumping rate* (*Jr*)[7] or *probability of opposition* (*po*)[12], which is a constant number in (0,1) and is often determined by empirical experiences. More descriptions about OBL can be found in the survey [14].

3 The Proposed Algorithm

3.1 Rotated-Based Explanation of OBL

In the OBL mechanism, an opposite number is mirrored to its original number by the center of the defined boundary in one-dimensional space (i.e. a number axis). However, a one-dimensional number axis can be placed in a two-dimensional space as well, so that the opposite number can be explained in another way.

As shown in Fig. 1(a), for a given two-dimensional plane with x axis and y axis, a number z, and its lower boundary a and upper boundary b, can be marked by points Z, A and B on x axis. The center point of the interval $[a, b]$ is denoted by C, its coordinate is $\frac{a+b}{2}$. Then, a circle can be drawn on the plane with the center of C, and the radius of $r(= \frac{b-a}{2})$. Through the point Z, a straight line,

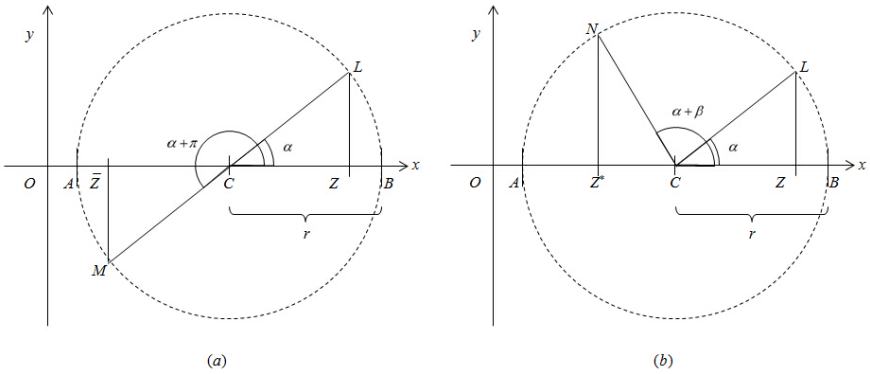


Fig. 1. The geometric interpretation of opposite number (a), and rotation number (b) in two-dimensional plane

which is perpendicular to x axis and intersects with the circle on point L, can be drawn. Obviously, the number z is the x position of point L, and the length of line segment $\overline{CL}$ is equal to the radius r of the circle. For the sake of convenience, two variables u and v are defined by:

$$\begin{aligned} u &= |\overrightarrow{C\bar{Z}}| = z - \frac{a+b}{2} \\ v &= |\overline{L\bar{Z}}| = \sqrt{r^2 - u^2} = \sqrt{(z-a)(b-z)}, \end{aligned} \tag{6}$$

where $|\overline{L\bar{Z}}|$ indicates the length of line segment $\overline{L\bar{Z}}$, and that $|\overrightarrow{C\bar{Z}}|$ denotes the length of the directed line segment $\overrightarrow{C\bar{Z}}$ together with the sign which indicates the direction of the line segment. Note that, the two points of the directed line segment should be on the same number axis in two-dimensional space.

Assume that the intersection angle of $\angle LCB$ is denoted by α, then the following formulas are satisfied.

$$\begin{aligned} \cos\alpha = \cos\angle LCB &= \frac{|\overrightarrow{C\bar{Z}}|}{|\overline{LC}|} = \frac{u}{r} \\ \sin\alpha = \sin\angle LCB &= \frac{|\overline{L\bar{Z}}|}{|\overline{LC}|} = \frac{v}{r} \end{aligned} \tag{7}$$

Let the point L rotate 180 degrees counterclockwise around the circle, a new point M will be reached. The projection of M on x axis is denoted by $\overline{Z}$. Obviously, its coordinate $\overline{z}$ is just the *opposite number* of z. Therefore, an opposite number in OBL can be explained as rotating 180 degrees of the original number in the two-dimensional plane.

3.2 Rotated-Based Learning (RBL)

Inspired from the analysis above, the OBL mechanism can be extended to a rotation-based learning (RBL) mechanism by rotating any specified deflection

Algorithm 1. The RBL mechanism for a population-based algorithm

1: {Update the boundary vectors a, b and its center point C for current population $P = \{Z_1, Z_2, \ldots, Z_D\}$}
2: **for** i =1 to D **do**
3: $a_i \leftarrow$ the minimum value of ith variable in P;
4: $b_i \leftarrow$ the maximum value of ith variable in P;
5: $c_i \leftarrow \frac{a_i+b_i}{2}$;
6: **end for**
7: {Produce the rotation-based population $RP = \{Z_1^*, Z_2^*, \ldots, Z_D^*\}$}
8: **for** $Z_i \in P$ **do**
9: {Generate the new rotation-based individual $Z_i^* = \{z_{i1}^*, z_{i2}^*, \ldots, z_{iD}^*\}$}
10: Set the rotation degree β;
11: **for** $j = 1$ to D **do**
12: Calculate the values of $u_{ij} \in u_i$ and $v_{ij} \in v_i$ by (10) and (11);
13: Calculate z_{ij}^* by (12).
14: **end for**
15: **end for**

angle. As shown in Fig. 1(b), for a given number z, the point Z, L, C, and the circle with radius r are all fixed. If rotating the L point by β degrees counterclockwise around the circle, then a new point N will be reached. So, the angle of $\angle NCB$ is $\alpha + \beta$. The projection of point N on x axis is point Z^*, its coordinate z^* on x axis is the **rotation number** of z. Let the quantity of directed line segment $\overrightarrow{CZ^*}$ be denoted by u^*, which is obtained by

$$u^* = r \times (\cos(\alpha + \beta)) = u \times \cos\beta - v \times \sin\beta. \tag{8}$$

Then, the **rotation number** z^* can be calculated by

$$z^* = (\frac{a+b}{2}) + u^*. \tag{9}$$

It is clearly that the rotation number can search any point in the rotation space through the rotation of different angles. Moreover, the concept of rotation number can be easily generalized to higher dimensions. Let $Z = (z_1, z_2, \ldots, z_D)$ be a vector with D variables. $a = [a_1, a_2, \ldots, a_D]$ and $b = [b_1, b_2, \ldots, b_D]$ are the lower boundary and upper boundary of Z, respectively, and $z_i \in [a_i, b_i]$, $i = 1, 2, \ldots, D$. Therefore, the center point of rotation space $[a, b]^D$ is indicated by $C = (c_1, c_2, \ldots, c_D)$, and $c_i = \frac{a_i+b_i}{2}$. The radius vector is $R = (r_1, r_2, \ldots, r_D)$, and $r_i = \frac{b_i-a_i}{2}$. Similar to (6), all the quantities of directed line segment $\overrightarrow{C_iZ_i}$ are indicated by the vector $U = (u_1, u_2, \ldots, u_D)$, and u_i is defined by

$$u_i = z_i - \frac{a_i + b_i}{2}. \tag{10}$$

The vector $V = (v_1, v_2, \ldots, v_D)$ is used to indicate all the lengths of point Z_i to its corresponding intersection point L_i on the circle, and v_i is defined by

$$v_i = \sqrt{(z_i - a_i)(b_i - z_i)}. \tag{11}$$

For every dimension, let the point Z rotate β degrees counterclockwise by the center point C, then the **rotation point** $Z^* = (z_1^*, z_2^*, \ldots, z_D^*)$ can be achieved, where the z_i^* is defined by

$$z_i^* = (\frac{a_i + b_i}{2}) + (u_i \times \cos\beta - v_i \times \sin\beta). \tag{12}$$

Based on the above analysis, the proposed RBL mechanism can rotate any degrees between 0° and 360° (same as the interval (−180°, 180°)), and explore any point in the search space. Actually, RBL mechanism also can be adapted to different application modes by specifying different angles. For example, when the deflection angle is fixed at 180 degrees, the RBL mechanism is equal to OBL mechanism. When the deflection angle is arranged carefully so that the rotation points can fall in between the center point M and the opposition point, RBL mechanism could be translated into the quasi-oppositional learning (QOL) mechanism [6]. So, the RBL mechanism can be seen as an extension of the OBL and QOL mechanisms. Moreover, if the deflection angle is set randomly among the interval (0°, 360°), then RBL can be seen as a random search. Therefore, the RBL mechanism is very flexible for finding the potential promising solutions.

Nevertheless, the Gaussian distribution is simply used in this paper to determine the deflection angle, and it is defined as

$$\beta = \beta_0 \cdot N(1, \sigma), \tag{13}$$

where β and β_0 indicate the deflection angle and its basic number, respectively. σ is the standard deviation of the Gaussian function $N(\cdot, \cdot)$, and is used to adjust the variation of the basic number. Before calculating the parameter, an initial value β_0 should be set firstly. In the following experiments, β_0 and σ will be set to 180° and 0.25, respectively. Then, β will vary mainly within [90°,270°], and fluctuate around the 180° randomly. This is also the key that RBL can overcome the disadvantages of the OBL mechanism.

Like OBL mechanism, the RBL mechanism also can be embedded into a population-based algorithm. In order to take RBL into practice, the rotation space should be obtained firstly by calculating the minimum and maximum values of each dimension of all individuals in the population. Moreover, the center point of the rotation space, the lower and upper bound vectors should be obtained as well. Then, the rotation-based population RP which consisting the rotation-based individuals generated by (10)–(12) will be produced. The implementation pseudocode of RBL mechanism is shown in Algorithm 1. Note that, the rotation angle β in step 10 should be generated by (13) in this paper.

3.3 Rotation-Based Differential Evolution

In order to verify the effectiveness of RBL mechanism, the similar algorithmic structure of ODE [7] is adopted to produce the rotation-based differential evolution (RDE) algorithm by embedding the RBL mechanism into DE algorithm.

Algorithm 2. The RDE Algorithm

```
Initialize the parameters pr_0 and β_0;
Randomly initialize each individual in population P;
Conduct RBL mechanism to produce the rotation-based population RP;
Select NP fittest individuals from {P, RP} as the initial population P;
while FEs ≤ MAX_FEs do
   if rand(0,1) ≤ pr then
      Conduct the RBL mechanism to produce the rotation-based population RP;
      Select NP fittest individuals from {P, RP} as the new population P;
      Update the pr by (13);
   else
      Execute the DE/rand/1/exp scheme;
   end if
   Update the best individual;
end while
Output the best individual;
```

The pseudocode of the RDE algorithm is shown in Algorithm 2. In RDE algorithm, the RBL mechanism is used for both population initialization and generation jumping. In population initialization phase, a basic population P is firstly initialized by randomly generating individuals. And then, the RBL mechanism shown in Algorithm 1 is used to produce the rotation-based population RP. Finally, all individuals in $P \bigcup RP$ are evaluated, and the NP best individuals are selected as the initial population P.

Similar to OBL mechanism, a new *probability of rotation* (pr) parameter, which is a real number within (0, 1), is introduced to control the execution of RBL mechanism. In each generation, if a random number belongs to (0,1) is less than pr, then RBL mechanism will be conducted to produce the rotation-based population. Then, the NP best individuals from $P \bigcup RP$ are selected as the new population for next generation. Otherwise, the classical $DE/rand/1/exp$ scheme (described in Section 2) will be executed to generate the next evolution-based population.

In RDE algorithm, the new probability of rotation (pr) parameter should be set ahead. For convenience, the self-adaptive mechanism defined in (13) is also used to generate the parameter pr. Using the self-adaptive mechanism, RDE algorithm only needs to predefine two initial values of parameters pr and β, which are indicated by pr_0 and β_0, respectively.

Compared with ODE algorithm, the main enhancements in RDE include the adoption of extended RBL mechanism, and the use of Gaussian distribution to generate the rotation angles. These two mechanisms can overcome the disadvantages of OBL mechanism which can only search one fixed point in the opposition space, and can improve the exploration ability of RDE algorithm. In addition, the self-adaptive mechanism also can improve the usability of RDE.

3.4 Time Complexy of RDE

Base on the analysis above, the algorithmic structure of RDE is very similar to ODE algorithm [7]. The main difference between them is the enhancement mechanism of DE algorithm, i.e. RBL or OBL mechanism. Nevertheless, the RBL also has the same computational time complexity with OBL mechanism. Therefore, same as ODE algorithm, computational time complexity of RDE algorithm is $O(T \cdot D \cdot O(F))$, in which T indicates the maximum fitness evaluation number, and $O(F)$ represents the computational time complexity of the optimization problems.

Table 1. Computational results achieved by OPSO, ODE, QODE, and RDE

Function	Dim	OPSO	ODE	QODE	RDE
F_1	30	**4.38E-62**	1.04E-28	3.59E-34	2.89E-31
F_2	30	**0.00E+00**	3.49E-09	**0.00E+00**	3.39E-14
F_3	30	**4.00E-08**	1.23E-01	4.92E-04	2.92E-02
F_4	30	2.07E+00	**3.24E-04**	6.67E-02	1.31E-02
F_5	30	**1.23E+01**	2.21E+01	2.28E+01	2.28E+01
F_6	30	1.33E-01	**0.00E+00**	1.00E-01	**0.00E+00**
F_7	30	2.88E-03	1.77E-03	**8.88E-04**	4.31E-03
F_8	30	-7.71E+03	-6.17E+03	-6.23E+03	**-1.26E+04**
F_9	30	4.61E+01	6.41E+01	1.05E+02	**1.02E-05**
F_{10}	30	1.17E+00	1.20E-14	**3.41E-15**	6.60E-15
F_{11}	30	2.08E-02	5.75E-04	2.71E-03	**0.00E+00**
F_{12}	30	2.60E-01	**3.02E-17**	**3.02E-17**	**3.02E-17**
F_{13}	30	1.07E-02	**2.88E-17**	3.66E-04	**2.88E-17**
$w/t/l$		8/0/5	7/3/3	6/1/6	–

"$w/t/l$" means that the RDE algorithm wins in w functions, ties in t functions, and loses in l functions to other algorithm.

4 Experimental Verifications

4.1 Test Functions

In the following experiments, 13 well-known and widely used global optimization functions are chosen as the test bed [1], [11]. Among these problems, the first five functions F_1 – F_5 are unimodal functions. F_6 is a step function which has a minimum and is discontinuous. F_7 is a noisy quartic function. F_8–F_{13} are multi-modal functions with many local minima.

Two groups of comparison will be taken in the following experiments. The first is between RDE and other three opposition-based algorithms, include ODE [7], QODE [6] and OPSO [4]. And the second comparison is between RDE and other state-of-the-art DE algorithms, include jDE [1], SaDE [5], JADE [16] and CoDE [13]. For the fair comparison, we follow the parameter settings and procedures specified of these algorithms in their original papers.

Based on the empirical experience, the parameters pr_0 in RDE algorithm is set to 0.2. Therefore, parameters pr will change in [0.1,0.3] but around 0.2 during the evolution. Some fixed parameters, such as NP=60, F=0.5 and Cr=0.9 are used for the proposed RDE algorithm. The maximum fitness evaluation number (MAX_FEs) is set to $5000\times D$. Each run stops when the MAX_FEs is met.

4.2 Comparison of RDE with Opposition-Based Algorithms

In order to verify the effectiveness of RBL mechanism, the comparison between RDE and some opposition-based algorithms is conducted firstly. The problem dimension D is set to 30. The experiment results and their rough comparisions are list in Table 1. It can be seen that the RDE algorithm obtains 6 best values among the 13 test functions, while the other three algorithms all get 4 best values. And that, RDE outperforms OPSO and ODE algorithms on 8 and 7 functions, respectively. Nevertheless, RDE and QODE all beat each other on 6 functions, and obtain the matched results.

Table 2 lists the results of the Friedman average ranking test among the four algorithms. It can be seen that RDE algorithm gets the least value, which means that RDE algorithm has the best comprehensive performance among the four algorithms. Therefore, RDE is a very competitive algorithm compared to other opposition-based algorithms. It also can be deduced that the RBL mechanism is more effective than traditional opposition-based learning mechanism.

Table 2. The average ranking achieved by RDE and other algorithms when D=30

Algorithm	RDE	QODE	ODE	OPSO
Ranking	**2.19**	2.46	2.62	2.73

4.3 Comparison of RDE with Some State-of-the-Art DE Algorithms

Experimental study is also conducted to compare the performance of the proposed RDE algorithm to other four algorithms, including jDE, SaDE, JADE and CoDE, on the mentioned test suite when D=50. Table 3 lists the experimental results achieved by the five algorithms. It can be seen that the proposed RDE algorithm achieves 7 best values, and three of them are optimum solutions. SaDE and jDE algorithm get 5 and 3 best values including 2 and 1 global optimums, respectively. In addition, JADE and CoDE algorithms all get only 1 best value.The comprehensive comparison results between RDE and other algorithms on the bottom of Table 3 show that, RDE outperforms CoDE, jDE, JADE and SaDE on 12, 9, 10 and 7 functions, respectively.

In addition, it can be seen that RDE algorithm gets 5 best values for the multi-model functions. Whereas, SaDE gets 4 best values on unimodal functions. This implies that the proposed RDE algorithm is better at solving the

Table 3. Computational results achieved by RDE and other four algorithms

Function	Dim	CoDE	jDE	JADE	SaDE	RDE
F_1	50	1.48E-04	3.46E-18	4.90E-55	**2.27E-69**	1.23E-30
F_2	50	5.84E-04	4.52E-11	7.82E-31	**5.77E-43**	1.63E-13
F_3	50	6.34E+04	2.45E+03	1.84E+02	**5.33E-04**	2.34E+01
F_4	50	3.44E+01	**8.04E-01**	5.15E+00	1.13E+01	1.42E+00
F_5	50	3.58E+02	4.06E+01	5.28E+01	**1.60E+01**	4.32E+01
F_6	50	**0.00E+00**	**0.00E+00**	2.40E+00	4.87E+00	**0.00E+00**
F_7	50	1.25E-01	1.59E-02	6.32E-03	1.04E-02	**6.03E-03**
F_8	50	-20656.70	-16154.80	-20933.40	**-20949.10**	**-20949.10**
F_9	50	6.89E+01	9.67E+01	**1.11E-09**	3.32E-02	7.21E-05
F_{10}	50	1.19E+01	4.58E-10	6.82E-01	1.25E+00	**9.56E-15**
F_{11}	50	1.14E-03	**0.00E+00**	8.03E-03	6.29E-03	**0.00E+00**
F_{12}	50	8.66E-04	1.87E-17	8.29E-03	2.90E-02	**1.81E-17**
F_{13}	50	7.19E-04	3.41E-17	1.46E-03	2.43E-01	**2.88E-17**
$w/t/l$		12/1/0	9/2/2	10/0/3	7/1/5	–

"$w/t/l$" means that the RDE algorithm is wins in w functions, ties in t functions, and loses in l functions to other algorithm.

Table 4. The average ranking achieved by RDE and other algorithms when D=50

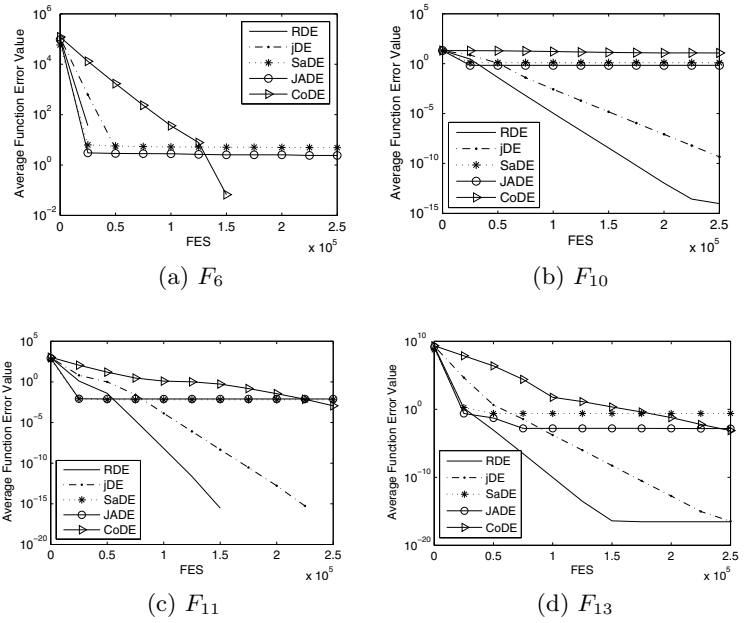
Algorithm	RDE	SaDE	jDE	JADE	CoDE
Ranking	**1.96**	2.92	2.96	3.00	4.15

Table 5. Wilcoxon's test between RDE and other algorithms when D=50

Algorithm	SaDE	jDE	JADE	CoDE
$p-$values	3.42E-01	6.15E-02	**2.87E-02**	**2.44E-04**

multi-modal problems. Fig. 2 illustrates the convergence curves achieved by the five algorithms on F_6, F_{10}, F_{11} and F_{13} when D=50. As seen that the convergence speed of RDE is very fast among the five algorithms.

The results of Friedman average ranking test are shown in Table 4. Obviously, RDE achieves the best comprehensive performance among the five algorithms. Table 5 shows the p-values applying Wilcoxon's test between RDE and the other four algorithms. The p-value below 0.05 (the significant level) are shown in **boldface**. As seen, RDE is significantly better than JADE and CoDE algorithms on the test bed when D=50. Although RDE is not significantly better than jDE and SaDE, but the ranking values in Table 4 show that RDE has better comprehensive performance among these algorithms.



(a) F_6 (b) F_{10}

(c) F_{11} (d) F_{13}

Fig. 2. The convergence curves of the five algorithms on F_6, F_{10}, F_{11} and F_{13}

5 Conclusions

In this paper, a novel rotation-based learning (RBL) mechanism is proposed by extending the OBL mechanism. Compared to the OBL, RBL can search any point in the rotation space of a number, so it is more flexible to find the promising solutions. By embedding the RBL mechanism into DE, the RDE algorithm is proposed. The comparisons between RDE and other opposition-based algorithms and some state-of-the-art DE algorithms demonstrate the effectiveness and efficiency of both the RBL mechanism and RDE algorithm.

In the future, the more effective application modes of RBL mechanism should be studied in order to improve the performance of RDE algorithms. Meanwhile, embedding the RBL mechanism into other nature-inspired algorithms will be investigated as well.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China under Grants (No.61364025 and No.61305150), the Humanity and Social Science Foundation of Ministry of Education of China (No. 13YJCZH174), the Foundation of State Key Laboratory of Software Engineering (No. SKLSE2012-09-19), and the Hebei Science and Technology Program Project (No. 12210319).

References

1. Brest, J., Greiner, S., Bošković, B., et al.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. 10(6), 646–657 (2006)
2. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Trans. Evol. Comput. 15(1), 4–31 (2011)
3. El-Abd, M.: Generalized opposition-based artificial bee colony algorithm. In: IEEE Cong. Evol. Compu., pp. 1–4. IEEE (June 2012)
4. Jabeen, H., Jalil, Z., Baig, A.R.: Opposition based initialization in particle swarm optimization (O-PSO). In: Proc. of the 11th Annual Conf. Comp. on Genetic and Evol. Comput., GECCO 2009, pp. 2047–2052. ACM (July 2009)
5. Qin, A.K., Huang, V.L., Suganthan, P.N.: Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 13(2), 398–417 (2009)
6. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Quasi-oppositional differential evolution. In: IEEE Cong. Evol. Compu., pp. 2229–2236. IEEE (September 2007)
7. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution. IEEE Trans. Evol. Comput. 12(1), 64–79 (2008)
8. Rahnamayan, S., Wang, G.G., Ventresca, M.: An intuitive distance-based explanation of opposition-based sampling. Applied Soft Computing 12(9), 2828–2839 (2012)
9. Storn, R., Price, K.: Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-012, International Computer Science Institute, Berkeley, CA (March 1995)
10. Tizhoosh, H.R.: Opposition-based learning: a new scheme for machine intelligence. In: Inter. Conf. Comput. Intell. for Model., Control and Auto., and Inter. Conf. Intell. Agents, Web Tech. and Inter. Commerce, vol. 1, pp. 695–701. IEEE (November 2005)
11. Wang, H., Rahnamayan, S., Sun, H., Omran, M.G.: Gaussian bare-bones differential evolution. IEEE Trans. Cyber. 43(2), 634–647 (2013)
12. Wang, H., Rahnamayan, S., Wu, Z.: Parallel differential evolution with self-adapting control parameters and generalized opposition-based learning for solving high-dimensional optimization problems. Journal of Parallel and Distributed Computing 73(1), 62–73 (2013)
13. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. 15(1), 55–66 (2011)
14. Xu, Q., Wang, L., Wang, N., Hei, X., Zhao, L.: A review of opposition-based learning from 2005 to 2012. Engineering Applications of Artificial Intelligence 29(1), 1–12 (2014)
15. Yang, X., Huang, Z.: Opposition-based artificial bee colony with dynamic cauchy mutation for function optimization. International Journal of Advancements in Computing Technology 4(4), 56–62 (2012)
16. Zhang, J., Sanderson, A.C.: JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In: IEEE Cong. Evol. Compu., pp. 2251–2258. IEEE (September 2007)
17. Zhou, X., Wu, Z., Wang, H., Li, K.: Elite opposition-based particle swarm optimization. Acta Electronica Sinica 41(8), 1647–1652 (2013) (in Chinese)