Diversity Analysis of Opposition-Based Differential Evolution—An Experimental Study

Hui Wang[1], Zhijian Wu[1], Shahryar Rahnamayan[2], and Jing Wang[1]

[1] State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, P.R. China
[2] Faculty of Engineering and Applied Science, University of Ontario Institute of Technology (UOIT), 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada
wanghui_cug@yahoo.com.cn, zjwu9551@sina.com, shahryar.rahnamayan@uoit.ca, wj.jxufe@gmail.com

Abstract. Opposition-based differential evolution (ODE) is a recently proposed DE variant, which has shown faster convergence speed and more robust search abilities than classical DE. The concept of opposition was utilized for the first time in optimization area to propose ODE. It is based on two important steps, generation jumping and elite selection. Some studies have pointed out that the first step improves diversity and provides more potential points to be searched (diversification), while the second step decreases diversity and accelerates convergence speed (intensification). However, there is not any experimental study to support this explanation. In this paper, we present an experimental study to analyze how the diversity changes in ODE. The experimental results confirm the explanation, and show that ODE makes a good balance between generation jumping and elite selection.

Keywords: Differential evolution (DE), Opposition-based DE, Opposition-based learning, Diversity analysis, Global optimization.

1 Introduction

Many real-world problems may be formulated as optimization problems with variables in continuous domains (continuous optimization problems). In the past decades, different kinds of nature-inspired optimization algorithms have been proposed to solve optimization problems, such as Simulated Annealing (SA) [1], Evolutionary Algorithms (EAs) [2], Particle Swarm Optimization (PSO) [3], and Differential Evolution (DE) [4], etc. According to frequently reported experimental studies [5], DE has shown better performance than many other nature-inspired algorithms in terms of convergence speed and robustness over several benchmark functions and real-world problems.

Since the DE algorithm is simple, efficient and easy to implement, it has attracted many researchers to work on improving its performance. Among different kinds of DE variants, opposition-based DE (ODE) [6] is one of the most excellent ones. The ODE employs an opposition-based learning (OBL) concept for opposition-based population initialization and generation jumping, in which current estimate and its

Z. Cai et al. (Eds.): ISICA 2010, LNCS 6382, pp. 95–102, 2010.

opposite estimate are considered at the same time in order to achieve a better approximation for a current candidate solution. It has been proved in [7], an opposite candidate solution has a higher chance to be closer to the global optimum solution than a random candidate solution. In [8], the OBL used in ODE is regarded as a diversity enhancement strategy to explain why ODE converges faster than classical DE. However, it did not give any experimental results to support the explanation. In this paper, we focus on analyzing the diversity of ODE based on experimental studies, and try to explain why ODE shows faster convergence speed and more robust search abilities than classical DE.

The rest paper is organized as follows. In Section 2, the opposition-based DE is briefly introduced. In Section 3, we analyze the diversity of ODE. Section 4 presents the test functions, experimental results and discussions. Finally, the work is concluded in Section 5.

2 Opposition-Based Differential Evolution

Opposition-based Learning (OBL) [9] is a new concept in computational intelligence, and has been proven to be an effective concept to enhance various optimization approaches [10], [11], [12], [13], [14]. When evaluating a candidate solution x to a given problem, simultaneously computing its opposite solution will provide a better chance to find another candidate solution closer to the global optimum.

Opposite Number [6]–Let $x \in [a, b]$ be a real number. The opposite of x is defined by:

$$x^* = a + b - x. \tag{1}$$

Similarly, the definition is generalized to higher dimensions as follows.

Opposite Point [6]–Let $X=(x_1,x_2,...,x_D)$ be a point in a D-dimensional space, where $x_1,x_2,...,x_D \in R$ and $x_j \in [a_j, b_j]$, $j \in \{1,2,...,D\}$. The opposite point $X^* = (x_1^*,x_2^*,...,x_D^*)$ is defined by:

$$x_j^* = a_j + b_j - x_j. \tag{2}$$

By applying the definition of opposite point, the opposition-based optimization can be defined as follows.

Opposition-based Optimization [6]–Let $X=(x_1,x_2,...,x_D)$ be a point in a D-dimensional space (i.e., a candidate solution). Assume $f(X)$ is a fitness function which is used to evaluate the candidate's fitness. According to the definition of the opposite point, $X^* = (x_1^*,x_2^*,...,x_D^*)$ is the opposite of $X=(x_1,x_2,...,x_D)$. If $f(X^*)$ is better than $f(X)$, then update X with X^*; otherwise keep the current point X. Hence, the current point and its opposite point are evaluated simultaneously in order to continue with the fitter one.

Similar to all population-based optimization, two main steps are distinguishable for DE, namely, population initialization and producing new generations by evolutionary operations such as mutation, crossover, and selection. ODE enhances these two steps using the OBL scheme. The original DE is chosen as a parent algorithm and the proposed opposition-based ideas are embedded in DE to accelerate its convergence speed.

For the population initialization, we implement this step as follows.

1) Randomly initialize the population P (N_p).
2) Calculate opposite population OP by

$$OP_{i,j} = a_j + b_j - P_{i,j}, \ . \tag{3}$$

where $i = 1, 2, ..., N_p$; $j = 1, 2, ..., D$, $P_{i,j}$ and $OP_{i,j}$ represent the jth variable of the ith individual of the population and the opposition population, respectively.

3) Select the N_p fittest individuals from $\{P \cup OP\}$ as initial population.

For the second step, ODE employs an opposition-based generation jumping as follows. By applying a similar approach to the current population, the evolutionary process can be forced to jump to a new solution candidate, which ideally is fitter than the current one. Based on a jumping rate J_r, the opposition is conducted on the current population after generating new populations by original DE operators. Then the N_p fittest individuals are selected from the union of the current population and the opposite population. Unlike the population initialization, generation jumping calculates the opposite population based on dynamic boundaries.

$$OP_{i,j} = \mathrm{MIN}_j^p + \mathrm{MAX}_j^p - P_{i,j}, \ . \tag{4}$$

where MIN_j^p and MAX_j^p are the minimum and maximum values of the jth dimension in current search space, respectively.

```
            Algorithm 1: ODE Algorithm
Begin
  Opposition-based Population initialization;
  while(BFV > VTR and NFC < MAX_NFC) do
    Execute classical DE algorithm;
    If (rand(0,1) < J_r) do
      Generate opposite population OP;
      Select N_p fittest individuals from P and OP as
      new population P;
    End if
  End while
End
```

The main steps of ODE are presented in Algorithm 1, where P is the current population, OP is the opposite population, $rand(0,1)$ is a uniform random number within [0,1], J_r is the jumping rate, BFV is the best fitness value so far, VTR is the value-to-reach [6], NFC is the number of evaluations, and MAX_NFC is the maximum number of evaluations.

3 Diversity Analysis of ODE

In DE, large population size N_p will improve the search abilities of DE to explore more potential regions. But this will slow down the convergence speed. In order to

increase the number of potential points to be searched while staying with a lower population size gives rise to the various strategies for diversity enhancement [8]. Opposition-based DE (ODE) is a typical example for diversity enhancement, which uses either the mutant vector obtained in the usual way or its opposite point based on a jumping rate J_r. Once the opposite population is generated, there are $2*N_p$ individuals available, N_p from the current population and another N_p from the opposite population. Based on an elite selection mechanism, the N_p fittest individuals are selected from those $2* N_p$ individuals to form the next generation population.

The ODE contains two important steps, generation jumping and elite selection. The first step can be regarded as a diversity enhancement mechanism, which increases the number of potential points to be searched. More potential points are generated to explore more regions. This will be helpful to improve the robustness of ODE. The second step is an elite section, which usually decreases diversity and speeds up convergence because only the best individuals are retained. Higher diversity can make convergence slower but it can increase robustness, while lower diversity can make convergence faster but it easily suffers from premature convergence. From the presented experimental results in [6], ODE obtains faster convergence speed and more robust search abilities than classical DE. It shows that ODE might make a good balance between generation jumping and elite selection. In this paper, we will focus on analyzing how ODE adjusts the diversity from the view of experimental studies.

4 Experimental Verification

4.1 Test Functions

In Rahnamayan's study [6], there were 58 benchmark functions used to verify the performance of ODE. In this paper, only the first 30 functions (f_1-f_{30}) are used (for the rest 28 functions, we got similar conclusions). For the descriptions of these functions, please refer to [6].

4.2 Results

In order to analyze the effects of the two steps, generation jumping (opposition) and elite selection, on the diversity, we divide the two steps into three states, before opposition, after opposition, and after selection. Then we calculate the diversity for each state, and observe the changes of diversity in these states. Let *DivBO*, *DivAO*, and *DivAS* represent the diversity of the three states, respectively (see Fig. 1).

The diversity of population is calculated as follows [15]:

$$Diversity(t) = \frac{1}{N_p} \sum_{i=1}^{N_p} \sqrt{\sum_{j=1}^{D} \left(X_{i,j}(t) - \overline{X_j(t)} \right)^2}, \tag{5}$$

where $X_{i,j}(t)$ is the jth value of the ith individual in population at generation t, and $\overline{X_j(t)}$ is defined by

DivBO: Diversity Before Opposition

DivAO: Diversity After Opposition

DivAS: Diversity After Selection

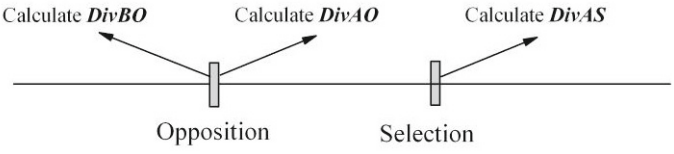

Fig. 1. Calculating diversity in different states

$$\overline{X_j(t)} = \frac{\sum_{i=1}^{N_p} X_{i,j}(t)}{N_p}. \tag{6}$$

Table 1 shows the average results of *DivBO*, *DivAO* and *DivAS* over the evolution on the 30 test functions (The *DivAO* is calculated for the double sized population, $2*N_p$). As seen, for all test functions, $DivAO > DivBO$ and $DivAS < DivAO$ are satisfied. The former inequality means that the diversity increases after generation jumping (opposition), and the latter one indicates that the diversity decreases after the elite selection. The first step is beneficial for diversification and improving the robustness, while the second step focuses on decreasing diversity and speeding up the convergence (intensification). That is why GODE shows faster convergence and more robust performance than classical DE.

Besides the average diversity analysis, we also calculate the probability of diversity increasing after generation jumping (*Prob_Div_Inc_AO*) and diversity decreasing after elite selection (*Prob_Div_Dec_AS*). The *Prob_Div_Inc_OP* and *Prob_Div_Dec_Sel* are computed as follows.

$$Prob\_Div\_Inc\_AO = \frac{Num\_Div\_Inc\_AO}{Num\_OP}, \tag{7}$$

$$Prob\_Div\_Dec\_AS = \frac{Num\_Div\_Des\_AS}{Num\_OP}, \tag{8}$$

where *Num_Div_Inc_AO* is the number of diversity increasing after generation jumping (it means that $Div\_AO > Div\_BO$), *Num_Div_Dec_AS* is the number of diversity decreasing after selection (it means that $Div\_AS < Div\_AO$), and *Num_OP* is the number of oppositions.

Table 1. The average results of *DivBO*, *DivAO* and *DivAS*

Functions	*DivBO*	*DivAO*	*DivAS*
f_1	1.268	**1.312**	1.189
f_2	1.191	**1.236**	1.114
f_3	16.949	**17.928**	16.795
f_4	0.0268	**0.02817**	0.02534
f_5	1.9893	**2.2335**	1.9658
f_6	146.15	**150.611**	137.30
f_7	1.3837	**1.4122**	1.3487
f_8	3.62709	**3.7891**	3.4074
f_9	1.05907	**1.23316**	0.9195
f_{10}	1.39196	**1.62557**	1.2717
f_{11}	21.1816	**21.6471**	19.457
f_{12}	0.163412	**0.174889**	0.1507
f_{13}	0.450783	**0.52986**	0.45443
f_{14}	1.13873	**1.34012**	1.1723
f_{15}	2.2847	**2.3771**	2.1457
f_{16}	49.656	**51.0923**	50.085
f_{17}	0.0684	**0.1137**	0.0689
f_{18}	0.157298	**0.1821**	0.1570
f_{19}	2.4949	**2.8431**	2.4886
f_{20}	3.1566	**3.9480**	3.243
f_{21}	1.132	**1.211**	1.092
f_{22}	18.958	**19.204**	17.628
f_{23}	68.5354	**70.5751**	64.149
f_{24}	0.19908	**0.20393**	0.19825
f_{25}	1.98192	**2.57376**	1.99292
f_{26}	2.08961	**2.40776**	1.90771
f_{27}	1.4323	**1.83425**	1.3224
f_{28}	1.41144	**1.6711**	1.29114
f_{29}	20.1733	**24.2225**	19.35
f_{30}	0.504458	**0.509575**	0.32736

Table 2 shows the average results of *Prob_Div_Inc_AO* and *Prob_Div_Dec_AS* over 30 runs. It can be seen that both the generation jumping and the elite selection increases and decreases diversity with very high average probabilities (more than 99%). The results demonstrate that the generation jumping always increases the diversity and provides more chances to explore more potential regions, while the elite selection could decreases the diversity each time and accelerate convergence speed. It also proves that $Div\_AO > Div\_BO$ and $Div\_AS < Div\_AO$ are satisfied.

Table 2. The average results of *DivBO*, *DivAO* and *DivAS*

Functions	*Prob_Div_Inc_AO*	*Prob_Div_Dec_AS*
f_1	100%	100%
f_2	100%	100%
f_3	94%	91%
f_4	100%	100%
f_5	100%	100%
f_6	100%	100%
f_7	100%	100%
f_8	100%	100%
f_9	89%	100%
f_{10}	100%	100%
f_{11}	100%	100%
f_{12}	100%	100%
f_{13}	100%	100%
f_{14}	100%	100%
f_{15}	100%	100%
f_{16}	100%	100%
f_{17}	100%	100%
f_{18}	100%	99.7%
f_{19}	100%	100%
f_{20}	100%	100%
f_{21}	100%	100%
f_{22}	100%	100%
f_{23}	100%	100%
f_{24}	100%	100%
f_{25}	100%	100%
f_{26}	93%	100%
f_{27}	100%	100%
f_{28}	100%	100%
f_{29}	100%	94%
f_{30}	100%	100%
Average	**99.2%**	**99.5%**

5 Conclusion

In this paper, we present an experimental study on diversity analysis of ODE. In ODE, there are two important steps, generation jumping and elite selection. The first step is beneficial for increasing diversity and exploring more promising regions, while the second one is helpful to speed up convergence. Although these two steps are incompatible, ODE makes a good balance between them towards searching candidate solutions. The experimental verification supports our explanations well. However, we only present an experimental study to analyze the diversity of ODE. That may not be enough to explain the advantages of ODE. Some theoretical analysis will be conducted in our future work.

Acknowledgments. This work was supported by the Jiangxi Province Science & Technology Pillar Program (No.: 2009BHB16400), and the National Natural Science Foundation of China (No.: 61070008).

References

1. Kirkpatrick, S., Gelatt, C.D., Vecchi, P.M.: Optimization by Simulated Annealing. Science 220, 671–680 (1983)
2. Back, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Publisher, New York (1996)
3. Storn, R., Price, K.: Differential Evolution–A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341–359 (1997)
4. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of International Conference on Neural Networks, vol. IV, pp. 1942–1948. IEEE Press, Piscataway (1995)
5. Vesterstrom, J., Thomsen, R.: A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems. In: Proc. Congress on Evolutionary Computation, Portland, vol. 2, pp. 1980–1987 (2004)
6. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-Based Differential Evolution. IEEE Transaction on Evolutionary Computation 12(1), 64–79 (2008)
7. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition versus Randomness in Soft Computing Techniques. Applied Soft Computing, 906–918 (2008)
8. Storn, R.: Differential Evolution Research – Trends and Open Questions. In: Chakraborty, U.K. (ed.) Advances in Differential Evolution. SCI, vol. 143, pp. 1–31 (2008)
9. Tizhoosh, H.R.: Opposition-Based Learning: A New Scheme for Machine Intelligence. In: Proceedings of International Conference on Computational Intelligence for Modeling Control and Automation, Vienna, Austria, pp. 695–701 (2005)
10. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-Based Differential Evolution Algorithms. In: Proceedings of Congress on Evolutionary Computation, pp. 2010–2017. IEEE Press, Vancouver (2006)
11. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: Opposition-Based Differential Evolution for Optimization of Noisy Problems. In: Proceedings of Congress on Evolutionary Computation, pp. 1865–1872. IEEE Press, Vancouver (2006)
12. Wang, H., Liu, Y., Zeng, S.Y., Li, H., Li, C.H.: Opposition-Based Particle Swarm Algorithm with Cauchy Mutation. In: Proceedings Congress on Evolutionary Computation, pp. 4750–4756. IEEE Press, Singapore (2007)
13. Rahnamayan, S., Wang, G.G.: Solving Large Scale Optimization Problems by Opposition-Based Differential Evolution (ODE). Transactions on Computers 7(10), 1792–1804 (2008)
14. Wang, H., Wu, Z.J., Rahnamayan, S., Kang, L.S.: A Scalability Test for Accelerated DE Using Generalized Opposition-Based Learning. In: Proceedings of International Conference on Intelligent System Design and Applications, Pisa, Italy, pp. 1090–1095 (2009)
15. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. Wiley & Sons, Chichester (2005)