Memetic Differential Evolution with Baldwin Effect and Opposition-Based Learning

Tong WENa*, Ji-xue HUA, Jin-shuai YANG and Xi-yang ZHAI
Air and Missile Defense College, Air-force Engineering University, Xi'an, China
a437424131@qq.com
*Corresponding author

Keywords: Memetic algorithm, Baldwin effect, Differential evolution, Hooke-Jeeves, Opposition-Based learning.

Abstract. To solve the current situation of immature use of Baldwin effect in existing memetic differential evolution (DE), we propose a memetic differential evolution with Baldwin effect and opposite-based learning (mDEBO), in which, Hooke-Jeeves and DE are combined into memetic DE algorithm through Baldwin effect. The individuals with better local search tend to be learned by others, which makes the population more diverse. In addition, the opposition-based learning mechanism is used to speed up the convergence rate. Compared with other well-known differential evolution algorithms in 15 benchmark functions in CEC2015, mDEBO performs satisfied convergence ability.

Introduction

Differential Evolution Algorithm [1], an important branch of evolution algorithm (EA), is popular for its simple codes, satisfied result, fast convergence and wide use. Improving its performance is a hotspot in the research in intelligent computation field. Standard DE and its improved versions have nice performance in many areas, which includes several algorithm competitions organized by IEEE CEC, during nearly 20 years since been proposed [2]. There are three major directions in the improvements of DE. Firstly, design new mutation strategy. JADE [3] replaces elite in population for sub-population in the mutation operator. Secondly, parameter self-adaptive mechanism and discrepant parameter set. [4] designs a mechanism which can learn the parameter set of succeed individuals in last generation. [5] and [6] control the parameter by making pre-attachment on parameter and generation, assisting with other state variables. [7] proposes a new mechanism with independent parameter set. Thirdly, hybrid with other scheme and algorithm such as Opposition-Based Learning [8], memetic algorithm [9], multi-population [10]. These improved methods improve DE’s performance in dealing with different issues.

Differential evolution algorithm for global search and other methods for local search operators combine together to be memetic differential evolution algorithm [9] [10] is an important branch of DE research, as well as memetic algorithm study. In memetic algorithm, how to choose and apply combination mechanism is an essential problem. Baldwin and Larmark learning, two types of typical combination mechanism of global search and local search, have been widely used in the improvement of other algorithms [11] [12].

In memetic GA (genetic algorithm) with Baldwin effect, individuals retain the fitness value of local search, while maintaining the original code unchanged, and then they can obtain larger survival probability in selection operation which is based on
roulette algorithm[13]. Similarly, in immune algorithm, the antibody retains the affinity obtained by local search and gives up the obtained code to achieve Baldwin effect. But in DE, the fitness value has little effect on deciding the evolution direction. If we still use the similar strategy in memetic GA or immune algorithm, which is to retain the fitness value obtained by local search and keep code unchanged, the effect would not be obvious. That’s why there are few scholars applying the Baldwin effect to the combination of DE and local search operator.

Based on the present research stated above, this paper proposes a memetic differential evolution algorithm with Baldwin effect and opposite-based learning, using a new method to achieve Baldwin effect in memetic DE. The algorithm chooses DE for global search, selecting superior individuals in the population to conduct Hooke-Jeeves, raises the selected probability of individuals who have better local search result, that is to achieve the Baldwin effect through increasing probability of being learned instead of keeping fitness value unchanged after the work of local search. Above operations will guide the population to potential directions. At the same time, the population has a certain probability to produce reverse solutions and conducts selection operation after evaluating the fitness value of all solutions to speed up the convergence rate. Compared with other excellent improved DE algorithms on 15 Benchmark functions in 30 dimensions, mDEBO performs better.

**Memetic DE with Baldwin Effect and Opposite-Based Learning**

**Hooke-Jeeves in mDEBO**

mDEBO adds the Hooke-Jeeves algorithm, which is also called pattern search, as a local search operator to the framework of memetic DE. In order to reduce the computational complexity, a simplified Hooke-Jeeves algorithm is designed in this paper, in which exploratory move and pattern move would conduct once in order on the individuals who deserve local search.

**exploratory move** Start from point \( x_k \), we move a step size of \( a \) in positive and negative on each axis at a time, then, retain the best individual in the detected points. Make \( x_{k+1} = x_k \) if \( f(x_{k+1}) < f(x_k) \) (for minimization), and conduct the pattern move; reduce the step size \( a \) to \( a/2 \) if \( f(x_{k+1}) > f(x_k) \), and repeat the exploratory move.

**pattern move** Move the point obtained from the exploratory move for as follow.

\[
x_{k+1} = x_k + b (x_{k+1} - x_k)
\]

where \( b \) is 1 in mDEBO.

The result of algorithm is related to the value of step size. In this paper, a self-adaptive parameter control strategy is proposed, which use successful step size to guide those in next generation. In mDEBO, the initial step size of exploratory move for each individual is as follows:

\[
a_{i,G} = \text{Norm} (m_{i,G}, S)
\]

where \( a_{i,G} \) is the step size of \( X_{i,G} \) which is the \( i \)th individual in \( G \)th generation, at each generation, it’s independently generated according to a normal distribution of mean \( m_g \) and standard deviation of 0.1. \( m_{i,G} \) is initialized at a value of 0.5 and updated as follows:

\[
m_{G+\text{learning period}} = wm_{G} + (1 - w) \text{mean} (S_G)
\]
where $w$ is a positive constant between 0 and 1 which can control the recursive speed and $\text{mean}(\cdot)$ represents the usual arithmetic mean of each component of $g$ to the 1.5th power, $S_g$ is the set of all successful $a$ at generation $G$.

**Baldwin Effect Learning Mechanism in mDEBO**

All of the individuals in [10] conduct local search, resulting in unnecessary computational cost. However, if only few elite individuals do local search, the performance of the algorithm will be affected slightly. Due to superior individuals can represent the development direction of population better than other individuals, superior individuals should do more local search than others, which would do help to reduce computational cost, as well as improve convergence speed.

Through experimental comparison and verification, we select superior individuals from the population for pattern search in accordance with the probability of $p_i(t)$.

$$p_i(t) = \begin{cases} 0, & f_i(t) < f_{\text{avg}}(t) \\ \sin((f_i(t) - f_{\text{avg}}(t))/p(f_{\text{best}}(t) - f_{\text{avg}}(t)) - p/2), & \text{otherwise} \end{cases}$$

$$f_i(t) = \frac{1}{f_{\text{old}}(t)} \times \text{individuals without local learning}$$

$$LR_i(t) = \frac{1}{f_{\text{old}}(t) + f_{\text{new}}(t)} \times \text{individuals with local learning}$$

In the next generation, mDEBO conducts mutation operation according to improved DE/rand/1:

$$V_{iG} = X_{r1G} + F_i(X_{r2G} - X_{r3G})$$

where $X_{r1G}$ is selected from whole population by roulette algorithm randomly.

In order to avoid too much computational cost or small improvement, mDEBO have local search every 4 generations, which is verified by experiments and would usually obtain satisfied results.

**Opposite-based Learning Strategy in mDEBO**

Opposite-based learning (OBL) is a new technology in the field of intelligent algorithm in recent years. It has been applied to DE [8] and other optimization algorithms. OBL uses the excellent properties of the reverse solutions to guide the evolution, which can apparently speed up the convergence rate without reducing the accuracy.

mDEBO adopts OBL strategy. We calculate the fitness value of each individual’s opposite solution after mutation, crossover and selection in each generation.

Define $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,j} \}$ as the phenotype of the individual $i$, and each of its opposite solution $X_i^{\text{Opt}} = \{x_{i,1}^{\text{Opt}}, x_{i,2}^{\text{Opt}}, \ldots, x_{i,j}^{\text{Opt}} \}$ is defined as follows.

$$x_{i,j}^{\text{Opt}} = a_j + b_j - x_{i,j}$$
where \( a_j = \min(x_{ij}) \) and \( b_j = \max(x_{ij}) \) is the dynamic boundary of population, which is helpful to preserve the historical experience and locate the opposite solution in the shrinking search space.

Then, we select the superior individuals for the collection of the original population and its reverse population as the new population.

**Steps of mDEBO**

For numerical optimization problems, the concrete implementation steps of mDEBO are as follows:

1. **Step 1** Initialize the relevant parameters. Distribute the population randomly and generate the opposite solutions of the population, and select the best NP individuals as the initial population.
2. **Step 2** Justify whether the termination criteria is satisfied.
3. **Step 3** Generate \( V_i \) based on the improved mutation operation.
4. **Step 4** Crossover and generate \( U_i \), evaluate the fitness values and retain successful search results and generate \( X_i' \).
5. **Step 5** Generate \( X_i^{opt} \) of the population and retain the best NP individuals.
6. **Step 6** Justify whether to meet the learning period, if not, return to step 2.
7. **Step 7** Select some individuals from the population according to \( p_i(t) \) to conduct Hooke-Jeeves algorithm.
8. **Step 8** Individuals change \( LR_i(t) \) according to the local search results.
9. **Step 9** Return to step 2.

**Experiments and Results**

**Experimental Setup**

In the experiments, we compare the performance of the proposed mDEBO with DE/rand/1 which is widely used, and three state-of-the-art DE, jDE [14], JADE [3], and CoDE [15] on the 15 benchmark functions used in CEC2015 [16]. NP is set to 100, the maximum number of function evaluation (Max_FES) is set to 10000, \( D \), \( F \) in mDEBO is set to 0.6, \( \text{CR} \) in mDEBO is set to 0.8, and other parameters of compared algorithms are same as their original literature. All algorithms are run by 51 times in 30 dimensions, the results are showed in Table 1. Figure 1 shows the 25th calculating progress of picked functions which are typical. All computations were carried out on a standard PC with Windows 7, Intel(R) Core(TM) i5-4590S CPU, 3.00GH, 3.00GHz, 4 GB RAM.

Algorithms with controlled restart described above are implemented in Matlab 2014a and this environment was used for experiments.

**Results and Comparison**

The results of mDEBO and compared algorithms are showed on Table 1. We can see the contrast of them clearly in the last line of the table. +/=/- means that mDEBO is superior/equal/ inferior to other DE for twice of the order of magnitude.

From Table 1, we can know mDEBO is superior to DE/rand/1/ JADE/ jDE/ CoDE in 14/ 13/12/14 benchmark functions, besides, not worse than them in any function.
Table 1. Experimental results of mDEBO and compared DE in 30 dimensions.

<table>
<thead>
<tr>
<th>Function</th>
<th>DE/rand/1</th>
<th>JADE</th>
<th>jDE</th>
<th>CoDE</th>
<th>mDEBO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean error (Std Dev)</td>
<td>Mean error (Std Dev)</td>
<td>Mean error (Std Dev)</td>
<td>Mean error (Std Dev)</td>
<td>Mean error (Std Dev)</td>
</tr>
<tr>
<td>F01</td>
<td>8.89E+04 (9.29E+04)</td>
<td>3.50E+02 (6.64E+02)</td>
<td>8.04E+04 (6.70E+04)</td>
<td>2.67E+04 (1.93E+04)</td>
<td>1.04E+04 (8.36E+03)</td>
</tr>
<tr>
<td>F02</td>
<td>4.46E-15 (1.04E-14)</td>
<td>2.17E-14 (1.22E-14)</td>
<td>2.79E-15 (8.54E-15)</td>
<td>6.09E+00 (2.59E+00)</td>
<td>2.21E-15 (3.72E-15)</td>
</tr>
<tr>
<td>F03</td>
<td>8.92E-15 (2.09E-14)</td>
<td>2.39E-03 (1.06E-02)</td>
<td>2.68E-14 (2.87E-14)</td>
<td>1.32E-04 (6.89E-05)</td>
<td>5.24E-18 (3.84E-18)</td>
</tr>
<tr>
<td>F04</td>
<td>4.34E-01 (6.90E-01)</td>
<td>9.31E+00 (2.57E+00)</td>
<td>1.11E+01 (4.47E+00)</td>
<td>2.16E+01 (1.92E+00)</td>
<td>3.43E-01 (5.14E-01)</td>
</tr>
<tr>
<td>F05</td>
<td>1.23E+02 (2.62E+01)</td>
<td>0.00E+00 (0.00E+00)</td>
<td>0.00E+00 (0.00E+00)</td>
<td>1.88E+01 (1.41E+00)</td>
<td>0.00E+00 (0.00E+00)</td>
</tr>
<tr>
<td>F06</td>
<td>2.36E+00 (2.28E-01)</td>
<td>2.62E-01 (3.80E-02)</td>
<td>4.35E-01 (5.78E-02)</td>
<td>9.82E-01 (1.34E-01)</td>
<td>6.85E-02 (4.98E-02)</td>
</tr>
<tr>
<td>F07</td>
<td>3.55E-01 (3.94E-02)</td>
<td>2.24E-01 (4.05E-02)</td>
<td>2.90E-01 (4.53E-02)</td>
<td>4.78E-01 (4.37E-02)</td>
<td>1.68E-01 (5.74E-01)</td>
</tr>
<tr>
<td>F08</td>
<td>2.78E-01 (2.91E-02)</td>
<td>2.40E-01 (3.90E-02)</td>
<td>2.92E-01 (3.11E-02)</td>
<td>2.87E-01 (2.96E-02)</td>
<td>6.47E-02 (5.97E-02)</td>
</tr>
<tr>
<td>F09</td>
<td>1.54E+01 (1.01E+01)</td>
<td>3.17E+00 (3.04E-01)</td>
<td>3.50E+02 (6.64E-02)</td>
<td>1.34E+01 (1.01E+00)</td>
<td>1.54E+00 (1.94E+00)</td>
</tr>
<tr>
<td>F10</td>
<td>1.38E+03 (1.77E+02)</td>
<td>2.27E+04 (1.26E+05)</td>
<td>2.26E+03 (3.44E+03)</td>
<td>1.43E+03 (2.04E+02)</td>
<td>6.61E+02 (9.12E+02)</td>
</tr>
<tr>
<td>F11</td>
<td>5.29E+01 (6.49E+00)</td>
<td>6.91E+01 (2.87E+01)</td>
<td>1.65E+01 (8.79E+00)</td>
<td>4.80E+01 (7.02E+00)</td>
<td>1.37E+01 (3.47E+01)</td>
</tr>
<tr>
<td>F12</td>
<td>4.57E+00 (2.95E-01)</td>
<td>4.44E+00 (7.72E-01)</td>
<td>4.61E+00 (7.58E-01)</td>
<td>7.19E+00 (8.16E-01)</td>
<td>5.49E-01 (9.74E-02)</td>
</tr>
<tr>
<td>F13</td>
<td>3.33E+01 (7.73E+00)</td>
<td>2.96E+03 (2.59E+03)</td>
<td>1.14E+01 (3.63E+00)</td>
<td>3.03E+01 (4.53E+00)</td>
<td>4.78E-03 (8.78E-04)</td>
</tr>
<tr>
<td>F14</td>
<td>6.76E+02 (1.25E+02)</td>
<td>1.76E+04 (5.48E+04)</td>
<td>2.33E+02 (1.65E+02)</td>
<td>7.28E+02 (1.34E+02)</td>
<td>4.67E+02 (1.22E+02)</td>
</tr>
<tr>
<td>F15</td>
<td>3.95E+01 (3.63E+01)</td>
<td>1.56E+02 (6.26E+01)</td>
<td>1.27E+02 (6.98E+01)</td>
<td>1.20E+02 (5.33E+01)</td>
<td>1.24E+01 (3.63E+01)</td>
</tr>
</tbody>
</table>

/+≈/- 13/2/0 12/3/0 11/4/0 14/1/0 14/1/0

Fig. 1 shows the operation of all the algorithms on 6 representative functions. We can see that the convergence rate of mDEBO is not inferior to other algorithms, and in most cases, the convergence rate is better than other algorithms.
Summary

In order to remedy the condition that little research on Baldwin effect in the combination of DE with local search algorithm, we try some new methods, which can improve the explore capability of population, and propose an efficient memetic DE algorithm. The introduction of opposition-based learning to the new algorithm can speed convergence observably. Tested on 15 benchmark functions in CEC2014 and compared with other state-of-art improved DE algorithms, mDEBO shows excellent capability.

References