Linear Weighted Gbest-guided Artificial Bee Colony Algorithm

Yanyu Zhang1,2,3, Peng Zeng1, Yang Wang1,3, Baohui Zhu1,3, Fangjun Kuang1,3
1. Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang China
2. School of Computer and Information Engineering, Henan University, Kaifeng China
3. Graduate School of Chinese Academy of Sciences, Beijing China
zyy@henu.edu.cn, zp@sia.cn, wangyang811023@ieee.org, zhubh@sia.cn, csukuangfj@126.com

Abstract — Artificial bee colony (ABC) algorithm invented recently by Karaboga is a competitive stochastic population-based optimization algorithm. However, solution search equation used in the original ABC algorithm is good at exploration but poor at exploitation. An improved ABC algorithm called Gbest-guided ABC (GABC) was introduced by researchers to improve the exploitation of ABC algorithm. In order to improve the GABC algorithm further, we propose an improved GABC algorithm with a linear weight called WGABC, and introduce a novel solution search equation used at scout bee stage of WGABC algorithm. Experimental results tested on a set of numerical benchmark functions show that WGABC can outperform ABC and GABC algorithms in most of the experiments.

Keywords — Artificial bee colony algorithm; Biological-inspired optimization algorithm; Numerical function optimization; Swarm intelligence

I. INTRODUCTION

In recent years, swarm intelligence becomes more and more attractive for researchers. Inspired by nature, people have proposed many swarm intelligence algorithms to solve optimization problems such as genetic algorithm (GA)[1-2], ant colony optimization (ACO)[3], artificial immune system (AIS)[4], particle swarm optimization (PSO)[5], and artificial bee colony algorithm (ABC)[6].

By simulating the foraging behavior of honey bee swarm, Karaboga invented ABC algorithm, a stochastic population-based algorithm, in 2005. Although ABC algorithm is a relatively new optimization method, it has received significant interest from researchers studying in different research areas and has been successfully applied to solve several practical problems due to its simplicity, ease of implementation and effectiveness after its invention. Comparative studies have demonstrated that ABC algorithm is a competitive optimization method compared to other algorithms such as GA and PSO [7].

ABC algorithm was originally proposed for solving numerical optimization problems [8]. Motivated by its success as a single-objective optimizer, researchers have extended the use of this algorithm to other areas. For example, Karaboga and Ozturk [9] applied ABC algorithm on training feed-forward neural networks to classify different data sets. Cobanli et al. [10] introduced a method based on ABC algorithm for active power loss minimization in electric power systems. Shukran et al. [11] developed a data mining algorithm based on ABC algorithm for classification tasks.

Similar to other population-based swarm intelligence algorithms such as PSO, the performance of ABC algorithm is determined by its exploration and exploitation abilities. The exploration refers to the ability of investigating various unknown regions in solution space to discover the global optimum, while the exploitation refers to the ability of finding better solutions using the knowledge of previous good solutions. The original ABC algorithm is good at exploration but poor at exploitation, which results in a low speed of convergence and easily getting trapped in the local optima when solving complex multimodal problems. To settle these problems, a number of variant ABC algorithms have been proposed. Inspired by improved strategies of PSO, Lei et al. [12] proposed an improved ABC algorithm, which introduced an inertial weight to the original ABC iteration equation to balance local and global searching processes. Zhu and Kwong [13] developed an improved ABC algorithm called gbest-guided ABC (GABC) algorithm by incorporating the information of global best (gbest) solution into the solution search equation to improve the exploitation, they tested the modified algorithm on a set of numerical benchmark functions, and experimental results showed that GABC algorithm can outperform the original ABC algorithm.

Inspired by [12-13], in order to improve the performance of GABC algorithm further, we modify the GABC algorithm by introducing a linear inertial weight into the solution search equation used at the employed bee and the onlooker bee stages to balance the exploration and exploitation processes. The inertial weight we proposed is different from the weight of [12] which is a nonlinear descending weight, while the inertial weight in this paper is linear increasing weight. Another improvement is that we replace the solution generating method used at the scout stage with a novel one. We name the GABC algorithm with a linear inertial weight as WGABC algorithm. In order to demonstrate the performance of WGABC algorithm, we test the algorithm on a set of benchmark functions, and experimental results show that WGABC algorithm is superior to ABC and GABC algorithms in most of the experiments.

The rest of this paper is organized as follows. Section II describes ABC algorithm briefly. Both GABC and WGABC algorithms are presented in Section III. Section IV presents and discusses experimental results. Finally, the conclusion is drawn in Section V.
II. ARTIFICIAL BEE COLONY ALGORITHM

As mentioned above, Karaboga [6] invented ABC algorithm for numerical function optimization by simulating the foraging behavior of honey bee swarm in 2005. ABC classifies foraging artificial bees into three groups, namely, employed bees, onlooker bees and scout bees. Employed bees are responsible for searching food around food sources in their memory and sharing the information of these food sources with onlooker bees through waggle dancing. By sharing the information of food sources with employed bees, onlooker bees waiting on the dancing area in the hive make decisions to choose food sources in probabilities which are proportional to the nectar amount of food sources. In ABC algorithm the number of employed bees is equal to the number of food sources, in other words, there is only one employed bee for one food source. An employed bee, whose food source has been exhausted by other bees, will become a scout. The scout bee searches randomly to find a new food source. By this mechanism, the exploitation will be handled by employed and onlooker bees while the exploration will be maintained by scout bee [14]. The framework of ABC algorithm [6] is summarized below:

Initialization

Repeat

Employed bee stage: update every solution in the solution population using (1).

Onlooker bee stage: Randomly select solutions based on their fitness values, then update them in the same way as at the employed bee stage.

Scout bee stage: Select one of the most inactive solutions, and replace it by a new randomly generated solution using (2).

Until (termination conditions are satisfied)

In ABC algorithm, a food source position represents a possible solution to the problem to be optimized and the nectar amount of a food source corresponds to the quality of the solution represented by that food source [6]. At initialization stage, ABC algorithm randomly generates SN solutions, where SN is the size of solution population. Each solution \( x_i (i=1,2,...,SN) \) is a \( D \)-dimensional vector. Here, \( D \) is the number of optimization parameters. After initialization, the population of the positions (solutions) is subjected to repeated cycles. At each repeated cycle, employed and onlooker bees produce new solutions from the old ones using (1).

\[
vt_j = x_{jt} + \alpha (x_{jt} - x_{kt}) + \beta (y_j - x_{jt})
\]

Here, \( k \in \{1,2,...,SN\} \) and \( j \in \{1,2,...,D\} \) are randomly chosen indexes.

One point should be pointed out is that \( k \) must be different from \( j \). \( x_{jt} \) (or \( v_{jt} \)) denotes the \( j \)th element of \( x_j \) (or \( v_j \)), and \( \psi_j \) is a uniform random number in \([-1, 1]\). At the scout bee stage, if a solution cannot be improved further through a predetermined number of cycles called \( \text{limit} \), an important parameter of ABC algorithm, then the solution is replaced by a new solution which is produced by a scout bee using (2).

\[
x_{jt} = x_{jmin} + \text{rand}(0,1)(x_{jmax} - x_{jmin})
\]

Here, \( x_j \) denotes the abandoned solution, and \( j \in \{1,2,...,D\} \) is a randomly generated index which denotes the modified dimension. \( x_{jmax} \) and \( x_{jmin} \) are \( j \)th dimension’s upper boundary and lower boundary, respectively. At each stage, every updated candidate solution is evaluated by artificial bees, and the performance of the new solution is compared with the performance of the old one. If the new solution is equal to or better than the old one, the new one replaces the old one in the memory. Otherwise, the old one is retained in the memory.

III. LINEAR WEIGHTED GBEST-GUIDED ABC ALGORITHM

As mentioned in section I, although ABC algorithm works well in solving optimization problem, there is still an insufficiency in ABC algorithm regarding its solution search equation. Equation (1) used to produce new solutions at both employed and onlooker bee stages is good at exploration but poor at exploitation. Both exploration and exploitation are necessary for ABC algorithm, but they contradict each other in practice. Well balancing these two abilities is a key to achieve a good optimization performance.

Zhu and Kwong [13] argued that the exploration ability of ABC algorithm is strong enough, and in order to improve the exploitation, they proposed a gbest-guided artificial bee colony algorithm which took advantage of the information of global best (gbest) solution to guide the search for candidate solutions. They modified the solution search equation that the original ABC algorithm used by adding a term called gbest term to the right-hand side of (1). The equation that they introduced is (3).

\[
v_{ij} = x_{ij} + \phi_j (x_{jt} - x_{kt}) + \psi_{ij} (y_j - x_{ij})
\]

Here, the first and second terms on the right-hand side of (3) are identical to those of (1). Symbol \( y_j \) is the \( j \)th element of the global best solution, and \( \psi_{ij} \) is a uniform random number in \([0, C]\), where \( C \) is a nonnegative constant. The constant \( C \) plays an important role in balancing the exploration and the exploitation of the candidate solutions searching. According to their study, the preferred value of \( C \) is 1.5.

Inspired by improved strategies of PSO, Lei et al. [12] proposed a modified solution search equation (4) to update the old solutions.

\[
v_{ij} = w v_{ij} + p (x_{jt} - x_{kj}) + q
\]

Here, \( w \) is a nonlinear decreasing weight defined by (5):

\[
w = 1 - \left( \frac{\text{iter}}{\text{maxiter}} \right)^n
\]

Here, \( \text{iter} \) is a current iteration number, \( \text{maxiter} \) is a maximal iteration number, and \( n \) is a positive constant determined by optimization function. They argued that five is a reasonable value of \( n \). \( p \) is defined as follows:

\[
p = p_0 \cdot \left( 1 - \frac{(\text{iter} - 1)}{\text{maxiter}} \right)
\]

Here, \( 0 < p_0 \leq 1 \) is a constant determined by algorithm designer. The last item \( q \) of (4) is a random disturbance strategy, which is defined by (7):
\[ q = q_0 \cdot (r - 0.5) \quad . \]  

Here, \( q \) is a small constant, usually within (0, 0.1) and \( r \) is a random number between (0, 1).

Inspired by [12], in order to balance the exploration and exploitation of GABC algorithm, we modified (3) through introducing a linear increasing inertial weight, and the solution search equation that we used to produce new solutions at employed bee and onlooker bee stages is defined by (8):

\[ v_{ij} = w_{ij}^e + \phi v_{ij} (x_{ij} - y_j) + w_{ij} (y_{ij} - y_j) \quad . \]

Here, \( w \) is the linear inertial weight we introduced, which is defined by (9):

\[ w = W_{\text{min}} + \frac{\text{iter}}{\text{maxiter}} (W_{\text{max}} - W_{\text{min}}) \quad . \]

In equation (8), \( 0 \leq W_{\text{min}} < W_{\text{max}} \leq 1 \), \( W_{\text{min}} \) and \( W_{\text{max}} \) are the minimal value and the maximal value of the inertial weight, respectively. Values of them are determined by algorithm developer. From (9), we can conclude that the inertial weight we introduced is different form the inertial weight in [12].

Other symbols in (8) are identical to symbols in (3). Based on the assumption that solutions of employed bees and onlooker bees will be far from the optimal solution in the first iterations, and it will converge closely to the optimal solution in later iterations. Equation (8) will dynamically adjust solutions of employed bees and onlooker bees by allowing them to wander with a wider step size in the solution space in the first iterations. The wandering step size will decrease as the number of the iteration increases.

The second improvement we made to GABC is that we replace (2) that GABC algorithm used to generate a new solution at the scout stage with (10):

\[ v_{ij} = x_{ij} + \rho_{ij} \left[ P_{\text{min}} + \frac{\text{iter}}{\text{maxiter}} (P_{\text{max}} - P_{\text{min}}) \right] x_{ij} \quad . \]

Here, \( 0 \leq P_{\text{min}} < P_{\text{max}} \leq 1 \), and the values of \( P_{\text{min}} \) and \( P_{\text{max}} \) represent the minimum and maximum percentage of the position adjustment for the scout bee[14]. Parameters are chosen by algorithm developer. Equation (8) shows that the exploration of GABC algorithm decreases with the advancement of the algorithm. Equation (10) implies that, with selected values of \( P_{\text{min}} \) and \( P_{\text{max}} \), the adjustment of the scout bee’s position based on its current position will linearly increase from \( P_{\text{min}} \times 100 \) percent to \( P_{\text{max}} \times 100 \) percent, and the exploration of the scout bee becomes from weak to strong. In one word, (10) can compensate the exploration loss of (8).

IV. EXPERIMENTS AND RESULTS

A. Benchmark functions

In order to demonstrate the performance of WGABC algorithm on numerical function optimization, six numerical benchmark functions used in [13] are used here. These functions are defined by (11-16) and their characters are listed in Table I.

\[ f_1(x) = \frac{1}{4000} \sum_{i=1}^{D} (x_i^2 - 10 \cos (\frac{x_i}{\sqrt{\sum_{j=1}^{D} x_j^2}})) + 1 \quad . \]

Here, every function’s global minimal value (or best value) is shown in the third column of Table I, and vectors listed in the fourth column are the corresponding vectors. Schaffer function, Griewank function and Ackley function are multimodal non-separable functions. Rosenbrock function is a unimodal non-separable function. Sphere function is a unimodal separable function, and Rastigrin function is a multimodal separable function.

B. Parameter Settings

In order to make the comparison based on an identical condition, the original ABC algorithm, GABC algorithm and WGABC algorithm are all implemented in VC 6.0 and ran on the same PC. There are three control parameters in the original ABC algorithm: the number of employed bees (SN), which is equal to the number of onlooker bees and the number of food sources, the value of limit and the maximum cycle number (MCN). In our experiments, the values of SN, limit and MCN were set to 40, 50 and 5000, respectively. The values of these parameters are identical to those in [13] expect parameter limit which is not specified in [13].

According to [13], the parameter \( C \) plays an important role in GABC algorithm. In order to investigate the effect of the parameter \( C \) on the performance of GABC algorithm, the authors of [13] tested GABC algorithm with different \( C \) on the benchmark functions, and finally they concluded that 1.5 is a preferable value of \( C \). So, in our experiments, the parameter \( C \) of GABC algorithm and WGABC algorithm is set to the same value.

In addition to parameters mentioned above, other parameters of WGABC algorithm are set as follows: the
values of \( W_{\text{min}} \), \( W_{\text{max}} \), \( P_{\text{min}} \) and \( P_{\text{max}} \) are set to 0.4, 1, 0.2 and 1, respectively.

C. Experimental results

Each experiment is repeated 30 times with different random seeds, and the mean and standard deviation of the output values of benchmark functions are recorded. Lower mean objective values indicate better solutions, and low standard deviation values mean better robustness of the algorithm. For each benchmark function, two kinds of dimensions of solution space were tested. Schaffer function and Rosenbrock function were tested with dimensions 2 and 3, and the other functions were all tested with dimensions 2 and 60.

Tables II - III show the optimization results of benchmark functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Algorithm</th>
<th>D=2</th>
<th>D=3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( \text{Mean} )</td>
<td>( \text{SD} )</td>
</tr>
<tr>
<td>Schaffer</td>
<td>WGABC</td>
<td>4.198078E-19</td>
<td>2.580282E-19</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>2.963038E-17</td>
<td>8.269320E-18</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
<td>5.836132E-04</td>
<td>2.414626E-04</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>WGABC</td>
<td>1.252809E-06</td>
<td>1.404101E-06</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>8.149000E-06</td>
<td>1.142859E-05</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
<td>1.965666E-03</td>
<td>1.899213E-03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Algorithm</th>
<th>D=30</th>
<th>D=60</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>( \text{Mean} )</td>
<td>( \text{SD} )</td>
</tr>
<tr>
<td>Sphere</td>
<td>WGABC</td>
<td>1.180558E-16</td>
<td>3.640863E-17</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>4.445634E-16</td>
<td>7.145910E-17</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
<td>1.030010E-15</td>
<td>2.485254E-16</td>
</tr>
<tr>
<td>Griewank</td>
<td>WGABC</td>
<td>1.110223E-16</td>
<td>2.465190E-32</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>4.810966E-16</td>
<td>5.967269E-17</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
<td>1.039099E-15</td>
<td>2.801099E-16</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>WGABC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
<td>7.626494E-14</td>
<td>2.313642E-13</td>
</tr>
<tr>
<td>Ackley</td>
<td>WGABC</td>
<td>6.516706E-15</td>
<td>4.733165E-30</td>
</tr>
<tr>
<td></td>
<td>GABC</td>
<td>3.197782E-14</td>
<td>3.442449E-15</td>
</tr>
<tr>
<td></td>
<td>ABC</td>
<td>1.146376E-13</td>
<td>2.778824E-14</td>
</tr>
</tbody>
</table>

One thing that needs to be noted is that some optimization results of GABC algorithm in Table II and Table III are a little different from those results in [13]. The reason is that parameter \( \text{limit} \), an important parameter influenced the performance of GABC algorithm, is not specified in [13], and the value of \( \text{limit} \) we used may be different from that in [13]. However, the comparison we made between WGABC algorithm and GABC algorithm in this paper is fair, because these two algorithms based on the same parameter settings and run on the same PC. From the results in Table II and Table III, it can be observed that the performance of WGABC algorithm is superior to the performance of GABC algorithm and the performance of the original ABC algorithm. When the benchmark function is Rosenbrock function with dimension 30, although the performance of WGABC algorithm is inferior to the performance of GABC algorithm, the difference between them is very slight. We have drawn the convergence curves of ABC, GABC and WGABC algorithms to show the progresses of the mean best values presented in Table II and Table III. For space limitation, only two representative cases.
of them are presented here, which are shown in Figs. 1 and 2, respectively.

![Convergence curves of ABC, GABC and WGABC algorithms for the Ackley function with dimension 60](image1)

![Convergence curves of ABC, GABC and WGABC algorithms for the Ackley function with dimension 30](image2)

From the results presented above, it can be seen that WGABC algorithm outperforms GABC algorithm, in other words, WGABC algorithm improves GABC algorithm greatly.

V. CONCLUSION

In this paper, an improved artificial bee colony (ABC) algorithm called Gbest-guided ABC (GABC) algorithm, which was proposed to solve the problem of ABC algorithm that the solution search equation of ABC algorithm is good at exploration but poor at exploitation, was studied. In order to improve GABC further, we proposed a linear weighted Gbest-guided artificial bee colony algorithm, called WGABC, and another improvement we made to GABC algorithm is that a novel solution search equation used at the scout stage was introduced to GABC algorithm. Experimental results tested on benchmark functions show that WGABC algorithm outperforms GABC and ABC algorithms.

ACKNOWLEDGMENT

This work is partly supported by the National High Technology Research and Development Program of China (863 Program: 2011AA040101, 2011AA040103), the Special Program for Key Basic Research Funded by MOST under contact 2010CB334705.

REFERENCES