Artificial Bee Colony Algorithm Based on Uniform Local Search

Yan Zhang(✉), Hu Peng, Changshou Deng, Xiaojing Wang, Haiyan Huang, and Xujie Tan

School of Information Science and Technology, Jiujiang University, Jiujiang, China
zy_xx_jju@163.com

Abstract. Although Artificial Bee Colony (ABC) algorithm is simple and efficient, it also has some disadvantages too. For example, the ABC is good at exploration but poor at exploitation and easily falls into local optimum. In order to overcome these shortcomings and improve the efficiency of the algorithm, the Uniform Local Search Artificial Bee Colony (UGABC) algorithm has been proposed in this paper. The algorithm greatly improves the exploitation ability. For the purpose of comparison, we used four algorithms to experiment. The experimental results show that the UGABC has the best accuracy and the fastest convergence rate among four algorithms.

Keywords: Artificial bee colony · Uniform design · Uniform local search · Gbest

1 Foreword

There are a large number of nonlinear, non-differentiable multi-peak complex optimization problems in the engineering technology and optimizations fields. Traditional optimization methods are difficult to solve these problems. In recent years, the intelligent algorithm proposed by scientists can effectively solve these complex problems. For example, Kenney et al. [1] proposed Particle Swarm Optimization (PSO) algorithm simulating birds predation behavior. Yang et al. [2] proposed Cuckoo Search (CS) algorithm simulating cuckoo parasitic brooding. In 2005, the Turkish scientist Karaboga [3] simulated the behavior of bee collecting honey proposed the Artificial Bee Colony (ABC) algorithm. Compared with some traditional evolutionary algorithms, ABC algorithm has the advantages of simplicity, high speed, strong performance, good robustness, etc. It has a very effect on continuous functions [4, 5], and has been widely supplied and developed.

It is crucial to strike a balance between local search and global search for algorithms to solve optimization problems. Emphasizing local search helps to improve the convergence speed of the algorithm, but it is easy to fall into local optimum. Emphasizing global search helps to find new optimal solutions and avoid premature convergence, but it will reduce the convergence speed of the algorithm. The ABC algorithm is better in global search, while the local search is slightly worse, which makes the algorithm easily fall into local optimum. Peng et al. [6] proposed a uniform

H. Peng et al. (Eds.): ISICA 2018, CCIS 986, pp. 17–26, 2019.
https://doi.org/10.1007/978-981-13-6473-0_2

local search method, which randomly selecting two individuals in the population, and generating a new optimal individual through uniform design, which can significantly enhance the local search ability of the algorithm, thereby improving the overall optimization of the algorithm performance.

The ABC algorithm is sensitive to the search strategy, and different search strategies significantly affect the optimization performance of the algorithm. Therefore, scholars have made a lot of improvements to the ABC algorithm's search strategy to improve the optimization performance of the algorithm. Inspired by the PSO algorithm, Zhu and Kwong [7] proposed the GABC algorithm, which introduces a Gbest in the search strategy, which improves the performance of the algorithm. Because the GABC algorithm has small changes to the search strategy, the effect is good and has received extensive attention.

Inspired by the GABC algorithm and the uniform local search method, we proposed a new algorithm named Uniform Local Search Gbest Artificial Bee Colony (UGABC) to solve the optimization problem. The UGABC algorithm combines the strong local search ability of ULS and GABC algorithms in order to improve the optimizations of the algorithm and solve the optimization problems.

2 ABC Algorithm and GABC Algorithm

2.1 ABC Algorithm

In the ABC algorithm, a Food Source represents a feasible solution of the problem to be solved. We use the "Fitness" to measure the pros and cons of a food sources. All bees are divided into Employed Bees, Onlooker Bees and Scout Bees. Different bees guide the entire bee colony to find quality food sources by sharing information and role conversion. At the beginning of the algorithm, all food sources are found by scouts, and then the food source is exploited by employed bees and scout Bees. Continued development has exhausted the resources of the food source, and the employed bees of the food source that depleted the resources are converted into scout bees to find more food sources. For convenience, we take the minimization problem as an example. The steps of the algorithm are as follows.

(1) Initialization phase: Randomly generate SN initial food sources.
(2) Employed bees search for new food sources in their respective food source neighborhoods, and choose a food source with a large fitness value using greedy method. When all employed bees complete the search, they returned to the dance area and share the information of food sources to the onlooker bees by means of swing dance.
(3) Onlooker bees select food sources based on information shared by employed bees, the greater the fitness value, the greater the probability of being selected.
(4) A food source is abandoned if the food source has not been updated after the Limit cycle. The corresponding employed bee is converted into a scout bee. The scout bees use the initialization formula to start randomly looking for new food sources.
(5) Record the best solution so far.

(6) Determine if the termination condition is met. If the termination condition is met, the optimal solution is output and the algorithm ends. Otherwise, go to (2).

In the step (2), use Eq. 1 to determine the neighbor food source.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{1}$$

Here x_{ij} is a randomly selected food source, φ_{ij} is a random number between [−1, 1], k is a randomly selected location index.

2.2 GABC Algorithm

The literature [7] proposed the GABC algorithm, which is based on the ABC algorithm to change the formula 1 into the formula 2. Although the changes are small, but the effect is good.

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) + \psi_{ij}(y_j - x_{ij}) \tag{2}$$

Here y_j is the Optimal solution of Column j, ψ_{ij} is a random number between [0, C]. C is a non-negative constant. C plays a very important role in balancing local search and global search. When C = 0, Eq. 2 becomes Eq. 1. From the literature [7] we know that when C = 1.5, the GABC algorithm works best.

3 Uniform Local Search Artificial Bee Colony Algorithm

3.1 Uniform Local Search

Uniform Design (UD) is an experimental design method jointly proposed by Professor Fang and mathematician Wang in 1978 [8]. The basic idea of UD is to use the number theory method to find some more uniform sets of points in the experimental area, and then use these points to arrange experiments. Such experimental results are representative and it can reduce the number of experiments. Literature [9] proved that if there are k factors, each factor has q levels, if a comprehensive experiment is performed; the number of experiments in the orthogonal design is q^2. The uniform design uses the uniform distribution theory to select q points to do experiments. So the number of experiments is q. When q is large, the superiority of uniform design is very prominent. Compared with the orthogonal design experimental method, the uniform design has the advantages of fewer experiments and better robustness.

Peng et al. proposed a Uniform Local Search (ULS) based on UD and applied it to the DE algorithm. The experimental results show that ULS can enhance the local search ability of the DE algorithm [6].

Like orthogonal design, uniform design also has a set of tables for building experiments. Generally, using $U_n(q^s)$ represent uniform design table. The table has n rows and s columns. Here n represents the number of experiments, s represents the maximum number of independent factors, and each factor contains n levels. Table 1 is a uniform design table. As you can see from Table 1, the maximum number of levels

per factor is equal to the number of experiments. For ease of use, uniform design provides a number of experimental tables, and the specific construction of the tables and more experimental tables can be found in Ref. [9, 10].

Table 1. Uniform design table $U_7(7^6)$

No.	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1
7	7	7	7	7	7	7

Literature [6] found that in the process of uniform local search optimization, the $U_6(6^6)$ uniform design table can be obtained by deleting the last row of $U_7(7^6)$. There are two reasons. First, the last line of $U_7(7^6)$ is the original individual, which is redundant. The second reason is that the experimental results obtained in advance show that the results of $U_6(6^6)$ is the best. In the experiment, each factor has six levels to get the best experimental results. If the number of levels of each factor is too large, it will take more evaluations to affect the performance of the algorithm [6].

The uniform local search step is as shown in Algorithm 1. In the ULS, if the problem dimension D is greater than six, we randomly decompose the D dimension into six groups. ULS requires six experimental individuals to be constructed. The total number of evaluations also needs to be six times.

Algorithm 1: Uniform local search

1: Input: Population SN, Function evaluation times FEs;
2: Randomly select two individuals $X_{i,G}$ and $X_{j,G}$ from SN;
3: Combining $X_{i,G}$ and $X_{j,G}$ construct six test individual Y_1,…, Y_6 according to $U_6(6^6)$;
4: Calculating the objective function value $f(Y_1)$,…, $f(Y_6)$;
5: Choosing an optimal individual O from Y_1,…, Y_6;
6: If $f(X_{i,G}) > f(\mathrm{O})$, replace $X_{i,G}$ with O;
7: $FEs = FEs + 6$;
8: Return and replace SN and FEs.

3.2 Uniform Local Search Artificial Bee Colony Algorithm Steps

The steps of UGABC are shown in Algorithm 2. We embedded the ULS into the loop of GABC in order to enhance the algorithm's local optimization ability. Although ULS can greatly improve the convergence speed, it is a greedy choice mechanism, so if the number of executions of ULS is too many, it may cause the algorithm to fall into local extreme. In order to strike a balance between convergence speed and population diversity, we only perform ULS once per cycle.

Algorithm 2: UGABC

1: Input the number of food sources SN , $Limit$ and Objective function evaluation times $MaxFEs$;
2: Randomly generate SN individuals;
3: Calculate the objective function value of each individual;
4: $FEs = SN$;
5: While $FEs < MaxFEs$
6: For $i = 1 : SN$
7:Eemployed bees to search for new food sources in the neighborhood according to formula (2) and record food sources with better fitness values;
8: Onlooker bees choose the food source to collect honey according to the information shared by the employed bees. The better the fitness value, the better the probability that the food source is selected;
9:If a food source has not been updated after a limit of cycles, then the food source will be abandoned by the employed bee. The employed bee becomes a scout bee. The scout bee uses the formula (2) to start randomly searching for new solutions;
10:Recording the optimal values and optimal solutions so far;
11:EndFor
12: execution algorithm 1 to perform Uniform Local Search;
13: EndWhile
14:Returns the optimal solution and the optimal value.

4 Experimental Results and Analysis

4.1 Test Function and Experiment Setup

In order to verify the effectiveness and superiority of the UGABC algorithm, we selected 13 commonly used benchmark functions in literature [11] as test set. In the simulation experiment, four algorithms of ABC, UABC, GABC and UGABC were selected for comparison experiments. Based on the principle of fairness, the parameters of these four algorithms are: SN = 50, Dim = 30, Limit = 50, MaxFEs = Dim * 5000, where C = 1.5 in GABC algorithm and UGABC algorithm, each benchmark function is independent run 30 times. The hardware environment used in the experiment was Intel I7 processor, 8 GB memory, and the software environment was Windows 7 and MATLAB 7.

In order to objectively and fairly evaluate the experimental results, the results were analyzed using Wilcoxon rank sum test and Friedman test in statistics. The Wilcoxon rank sum test is based on the rank sum of the samples to determine whether the two samples are from the same population. The Wilcoxon rank sum test can analyze whether there is a significant difference between the algorithm participating in the comparison and the experimental result of the UGABC algorithm running independently 30 times on the benchmark functions. The Friedman test uses rank to analyze whether there is a significant difference in the population distribution of multiple independent samples. The Friedman test ranks the rank mean of each sample. The smaller the rank means, the better the test results [12].

4.2 UGABC Algorithm Quality Analysis

The average error and standard deviation of the UGABC algorithm and the other three algorithms are shown in Table 2. The significance level of the Wilcoxon rank sum test is 0.05. At the bottom of Table 2, the symbols "−", "+", and "≈" are used to indicate that the corresponding algorithm is inferior, superior, and equivalent to the UGABC algorithm. The rank of Friedman test is in Table 4. Figure 1 plots the average convergence curve for each algorithm running independently for 30 times on 13 benchmark functions. It can be found from Table 2 that the UGABC algorithm has strong convergence ability and good convergence precision. From the results of the Wilcoxon rank sum test in the last three rows of Table 2, it can be seen visually that the UGABC algorithm is the best among the algorithms involved in the comparison.

Table 2. Mean error value and standard deviation of algorithms and comparison results based on Wilcoxon's rank sum test

F	MeanError ± StdDev			
	ABC	UABC	GABC	UGABC
f_1	1.54E−16 ± 2.15E−16−	3.67E−23 ± 5.00E−23−	4.43E−32 ± 3.80E−32−	6.17E−38 ± 4.42E−38
f_2	8.85E−11 ± 3.40E−11−	1.62E−13 ± 9.05E−14−	6.34E−18 ± 2.02E−18−	1.40E−20 ± 6.48E−21
f_3	8.24E+03 ± 1.55E+03−	9.06E+01 ± 4.91E+01≈	9.36E+03 ± 2.29E+03−	9.67E+01 ± 4.60E+01
f_4	4.73E+01 ± 4.30E+00−	2.74E+01 ± 3.79E+00−	3.72E+01 ± 4.48E+00−	1.89E+01 ± 1.96E+00
f_5	8.48E−01 ± 6.38E−01+	7.30E−01 ± 1.04E+00+	7.99E−01 ± 3.02E+00+	1.57E+01 ± 2.61E+01
f_6	1.67E−01 ± 3.723E−01−	0.00E+00 ± 0.00+00≈	0.00E+00 ± 0.00+00≈	0.00E+00 ± 0.00+00
f_7	3.08E−01 ± 5.08E−02−	8.66E−02 ± 1.57E−02−	1.37E−01 ± 2.18E−02−	4.44E−02 ± 1.28E−02
f_8	3.82E−04 ± 2.97E−09−	3.82E−04 ± 5.46E−11−	3.82E−04 ± 2.21E−08−	3.82E−04 ± 4.54E−13
f_9	4.07E−12 ± 1.90E−11−	8.70E−15 ± 1.73E−14−	4.14E−16 ± 1.09E−15−	0.00E+00 ± 0.00+00
f_{10}	3.95E−10 ± 1.83E−10−	6.04E−12 ± 2.92E−12−	5.02E−14 ± 7.21E−15−	3.19E−14 ± 3.61E−15
f_{11}	1.08E−07 ± 5.83E−07+	7.40E−18 ± 2.77E−17≈	4.77E−06 ± 2.57E−05+	4.92E−04 ± 2.65E−03
f_{12}	4.50E−19 ± 6.19E−19−	7.85E−25 ± 8.10E−25−	1.58E−32 ± 1.98E−34−	1.57E−32 ± 5.47E−48
f_{13}	1.36E−16 ± 2.91E−16−	2.69E−23 ± 3.81E−23−	3.47E−32 ± 1.63E−32−	1.35E−32 ± 5.47E−48
−	11	9	10	–
+	2	1	2	–
≈	0	3	1	–

As can be seen from Fig. 1, the UGABC algorithm has stronger convergence ability than other algorithms. The convergence speed of UGABC is slightly worse than other algorithms in f_5 and f_{11}. In addition to this, the UGABC algorithm has an absolute advantage. Table 3 gives the Friedman test rankings for the UGABC algorithm and the algorithms involved in the comparison. The GUABC algorithm ranks first, which illustrates the advantages of the UGABC algorithm from a statistical perspective.

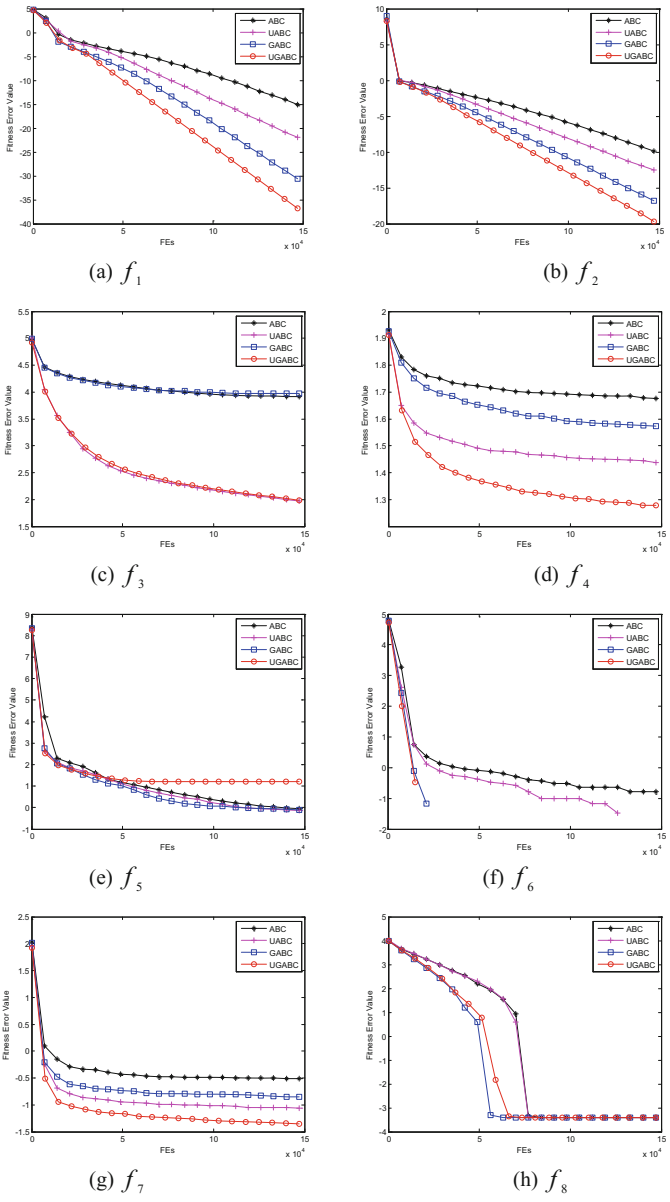
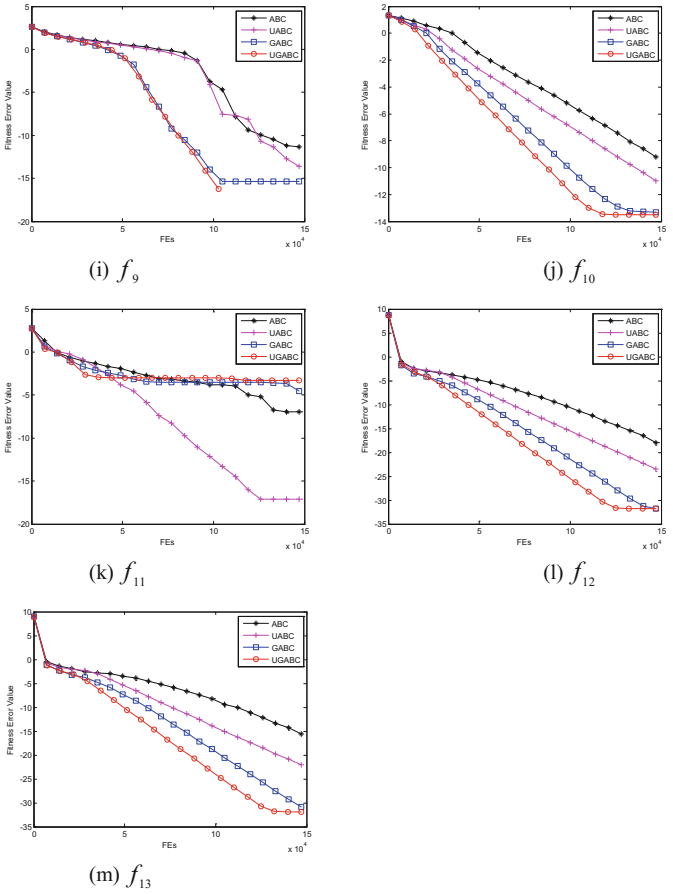


Fig. 1. Convergence curve on four algorithms on benchmark functions



(i) f_9 (j) f_{10}

(k) f_{11} (l) f_{12}

(m) f_{13}

Fig. 1. (*continued*)

Table 3. Average rankings achieved by Friedman test

ABC	UABC	GABC	UGABC
3.62	2.19	2.54	1.65

4.3 Dimension Change Analysis

To further illustrate the advantages of the UGABC algorithm, we selected different dimensions for experimentation and recorded experimental results. Table 4 gives the experimental results of the four algorithms in 50 dimensions. Similar to Table 2, in order to objectively compare the advantages and disadvantages of the algorithm, the last three lines of Table 4 gives the Wilcoxon rank sum test results of four algorithms. Table 5 gives the Friedman test results for the 50-dimensional.

From Table 4, we find that in the 50-dimensional problems, the UGABC algorithm is significantly better than the ABC algorithm in all benchmark functions. Compared with the UABC algorithm, UGABC has the comparable experimental results on both f_5 and f_{11}. UGABC algorithm is superior to UABC in the rest of the benchmark functions. Compared with the GABC, UGABC slightly worse than GABC in f_5. Both algorithms converge to 0 in f_6. UGABC's test results are better than GABC in other functions.

Table 4. Mean error value and standard deviation of algorithms at dim = 50 and comparison results based on Wilcoxon's rank sum test

F	MeanError ± StdDev			
	ABC	UABC	GABC	UGABC
f_1	2.75E−15 ± 4.37E−15−	4.19E−19 ± 7.77E−19−	1.40E−30 ± 1.82E−30−	3.08E−39 ± 2.07E−39
f_2	2.87E−10 ± 1.38E−10−	8.31E−12 ± 7.06E−12−	3.53E−17 ± 8.03E−18−	3.30E−21 ± 1.48E−21
f_3	3.03E+04 ± 2.84E+03−	5.48E+02 ± 1.80E+02≈	3.41E+04 ± 6.60E+03−	6.56E+02 ± 2.50E+02
f_4	7.04E+01 ± 3.15E+00−	4.14E+01 ± 2.27E+00−	6.61 E+01 ± 3.31E+00−	3.84E+01 ± 2.68E+00
f_5	1.48 E+01 ± 8.82E−01−	1.74E+01 ± 2.80E+01≈	5.61E+00 ± 1.54E+01+	2.81E+01 ± 3.55E+01
f_6	2.23E+00 ± 1.28E+00−	1.17E+00 ± 6.87E−01−	0.00E+00 ± 0.00+00≈	0.00E+00 ± 0.00+00
f_7	8.29 E−01 ± 1.14E−01−	2.09 E−01 ± 5.09E−02−	3.91 E−01 ± 7.16E−02−	1.06 E−01 ± 2.81E−02
f_8	6.37 E−04 ± 6.60E−08−	6.36 E−04 ± 3.49E−10−	6.36 E−04 ± 1.09E−08−	6.36 E−04 ± 2.24E−12
f_9	2.51E−08 ± 1.31E−07−	5.88E−13 ± 2.06E−12−	1.77E−14 ± 2.84E−14−	0.00E+00 ± 0.00+00
f_{10}	6.47E−10 ± 2.23E−10−	1.11E−09 ± 8.89E−10−	1.29E−13 ± 1.45E−14−	6.56E−14 ± 6.72E−15
f_{11}	9.81E−12 ± 5.04E−11−	8.62E−05 ± 4.62 E−04−	1.97E−10 ± 1.05E−09−	0.00E+00 ± 0.00+00
f_{12}	1.45E−18 ± 1.61E−18−	1.12E−20 ± 1.79E−20≈	1.17E−32 ± 3.52E−33−	9.42E−33 ± 8.69E−36
f_{13}	1.01E−15 ± 9.14E−16−	1.59E−19 ± 2.28E−19−	6.24E−31 ± 5.96E−31−	1.35E−32 ± 2.21E−34
−	13	11	11	–
+	0	0	1	–
≈	0	2	1	–

Table 5. Average rankings achieved at dim = 50 by Friedman test

Dim	ABC	UABC	GABC	UGABC
50	3.46	2.73	2.42	1.38

5 Conclusion

In order to improve the local search ability of the ABC algorithm, this paper proposes an artificial bee colony algorithm based on uniform local search. The algorithm introduces Gbest for neighborhood search in the stage of employed bees and onlooker bees, and embeds ULS after each cycle, which significantly improves the local search ability of the algorithm. It can be seen that the UGABC algorithm is superior to the comparison algorithm in solving accuracy, convergence speed and running time from the test results of different dimensions of 13 standard benchmark functions.

Acknowledgement. This work was supported by The National Science Foundation of China (No. 61763019), The Natural Science Foundation of Heilongjiang Province (General Program: F2017019), The Science and Technology Plan Projects of Jiangxi Province Education Department (No. GJJ161072, No. GJJ161076, No. GJJ170953), The Education Planning Project of Jiangxi Province (No. 15YB138, No. 17YB211).

References

1. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 4th IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Service Center, Piscataway (1995)
2. Yang, X.S., Deb, S.: Cuckoo search via Levy flights. In: World Congress on IEEE Nature & Biologically Inspired Computing, pp. 210–214 (2009)
3. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. TR06, Computers Engineering Department, Engineering Faculty, Erciyes University, Kayseri (2005)
4. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **2**(14), 108–132 (2009)
5. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. **8**(1), 687–697 (2008)
6. Peng, H., Wu, Z., Deng, C.: Enhancing differential evolution with communal learning and uniform local search. Chin. J. Electron. **26**(4), 725–733 (2017)
7. Zhu, G., Sam, K.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. **217**(7), 3166–3173 (2010)
8. Wang, Y., Fang, K.: A note on uniform distribution and experimental design. Mon. J. Sci. **26** (6), 485–489 (1981)
9. Fang, K.: Uniform design-number theory method in the application of experimental design. Acta Math. Appl. Sinica **04**, 363–372 (1980)
10. Fang, K.: Uniform design. Tactical Missile Technol. **02**, 56–69 (1994)
11. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)
12. Peng, H., Wu, Z., Zhou, X., et al.: Bare-bones differential evolution algorithm based on trigonometry. J. Comput. Res. Dev. **12**, 2776–2788 (2015)