Artificial bee colony Programming Made Faster

Xingbao Liu[1,2], Zixing Cai[1]
1. School of Information Science and Engineering, Central South University, Changsha 410083, China
2. Center of Modern Education Technology, Hunan Business College, Changsha 410025, China.
liuxb0608@gmail.com

Abstract

The artificial bee colony (ABC) algorithm is a stochastic, population-based evolutionary method that can be applied to a wide range of problems, including global optimization. The paper proposes a variation on the traditional ABC algorithm, called the artificial bee colony programming, or ABCP, employing randomized distribution, bit hyper-mutation and a novel crossover operator to significantly improve the performance of the original algorithm. Application of the new ABC algorithm on fifteen benchmark optimization problems shows a marked improvement in performance over the traditional ABC..

1. Introduction

The bee colony foraging behaviors demonstrates some intelligent characteristics. Inspired by the nature swarm behavior, D.Karaboga proposed a simple yet powerful population-based algorithm, artificial bee colony (ABC), to solve the numerical optimization problems [1]. ABC algorithm has gradually become more popular and has been widely used to many practical cases, mainly because it showed good convergence properties [2].

Some useful models have been constructed to simulate the intelligent foraging behaviors of bee colony and applied for solving combinatorial problems [3,4]. Yang proposed a virtual bee algorithm (VBA) to solve the numerical optimization problems[5]. In VBA, a swarm of virtual bees are generated randomly and move randomly in the state space. These virtual bees interact when they find better food source corresponding to a better function value. The optimum of solving problem can be obtained by the density of interaction between virtual bees. But VBA can only optimize some special functions with two parameters. In order to overcome the limitation, D.Karaboga developed an artificial bee colony algorithm and applied it to high dimensional multi-modal function optimization problems. In the ref.[2], D.Karaboga compared the performance of ABC algorithm with that of differential evolution (DE), particle swarm optimization (PSO) and evolutionary algorithm (EA) for high-dimensional benchmarks.

2. Artificial bee colony algorithm

In ABC algorithm the colony of artificial bee contains three groups of bees: employed bees, onlookers and scouts. Employed bees are placed randomly in the food source; onlookers are waiting in the dancing area for making decision on which food sources to be chosen, while the scouts carry out random search. The process of the algorithm is similar to the artificial immune algorithm based on clone selection principle; and the main steps of the algorithm are given below [2]:

- ➢ Initialization
- ➢ Repeat
  - ✧ Place employed bees on the food sources
  - ✧ Place onlookers on the food sources
  - ✧ Send scout to carry out the global search.
- ➢ Until the terminal condition are satisfied.

In the above ABC algorithm, every cycle contains three stages. First of all employed bees are initialized randomly and then to be evaluate their fitness, which is a common process just like DE, EA and PSO do. In the next stage, employed bees return to their previous food source and go on exploring the neighborhood of the source; meanwhile, every onlooker chooses a food source offered by employed bees according to its fitness, and start to explore its neighborhood. During the neighborhood exploration, the employed bee would be substituted by the bee better fitness. In the last stage, the scouts are sent to possible food source.

[1] This work was supported in part by the National Natural Science Foundation of China under Grant 90820302 and 60805027, and in part by the Research Fund for the Doctoral Program of Higher Education under Grant 200805330005.

An onlooker bee select a food source depending on the probability value associated with that food source, p_i obtained by the formation (2.1)

$$p_i = \frac{fit(eb_i)}{\sum_{i}^{SN} fit(eb_i)} \tag{2.1}$$

Where, $fit(eb_i)$ is the fitness of ith employed bee, SN is the size of population. In basic ABC algorithm, the group of employed bees is evolutionary population, and its size is equal to the onlookers group.

For reproducing the new food source from the employed bees, onlookers carry out a mutation operator, and become the candidate food source around the corresponding employed bees. The ABC algorithm employs a bit mutation operator which is similar to differential mutation operator. the formulation is listed as formulation (2.2)

$$V(i,j) = eb(i,j) + \xi_{ij} \times (eb(i,j) - eb(k,j)) \tag{2.2}$$

Where the parameter ξ is a random number from -1 to 1, and $k \in \{1,2,\cdots,NP\}$, $k \neq i$, $j \in \{1,2,\cdots,D\}$ and D is the amount of function dimension. It produce candidate food source around $eb(i,j)$.

For detailed principle and explanations behind parameters settings, evolutionary strategy, and even simulated results, please refer to[1,2].

3. Artificial bee colony programming

From the above analysis to the artificial bee colony principle, we conclude that the ABC algorithm focuses on the neighborhood search, while its global search capability is weak. This mainly manifests the following two aspects.

Onlookers select food source completely depend on the fitness of the corresponding employed bees. Under the strategy, those individuals with better fitness have large chances to get more onlookers. After the mutation operator the modified onlooker bees carry out the neighborhood search. But those employed bees with worse fitness only obtain few onlookers during the greedy selection strategy, so their neighborhoods are not explored enough and discarded finally. What's more, the diversity information of population is lost, and the evolutionary process could fall into local optimum.

The scouts are up to global search task through distributing food sources randomly, but the amount of scouts is about 10% to the population size, so the global search capability of ABC algorithm is poor. On the other hand, the strategy of global search is only randomly distribution, which is too simple to work for practical problem.

In order to strengthen the global search capability and maintain diversity information in ABC algorithm, two crucial issues in proposed ABCP are addressed, that is, modifying the greedy selection scheme and applying a novel crossover operator.

3.1. Randomized selection strategy

Choosing suitable selection scheme frequently is necessary to keep diversity information. In ABCP, we propose a random strategy for the selection process. Firstly we increase the amount of onlookers, though the amount is equal to the size of employed bees in basic ABC algorithm. Secondly we discarded the greedy selection scheme and adopted a randomized distribution strategy, and which is not depended on the fitness of employed bees. The following formulation (3.1) shows the distribution strategy in detailed.

$$onlook = \lfloor NP \times Rand(NP,1) \rfloor \tag{3.1}$$

Where, NP is the size of employed bee group; $Rand$ is a stochastic function, and the expression $Rand(NP,1)$ generates NP scalar randomly, and every scalar satisfies (3.2).

$$\forall x \in Rand(NP,1), x \in [-1,1] \tag{3.2}$$

The Eq.(3.1) generates a vector with length NP. After the operation, employed bee obtained $onlook(i)$ onlookers.

Meanwhile, ABCP utilizes the same mutation strategy, and for details, please refer to Eq. (2.2). The table 2 shows the framework of neighborhood search in ABCP.

Table 1. The neighborhood search scheme in ABCP

```
1.  Sort employed bees set EB randomly.
2.  distribute the onlookers randomly as Eq.(3.2),
    and get the distribution vector Onk
3.  for every individual in EB
4.      select individual p in EB randomly,
5.      For k=1:1:Onk(i)    % explore the
    neighborhood of EB(i)
6.          For j=1:1:D
7.              generate a random number w
8.              m（i,j）=EB(i,j)+w*(EB(i,j)-q(i,j))
9.          calculate the fitness of m;
10.         if m is better than EB(i), then x<—m;
11.     end
12. end
```

3.2. Layer Noisy Crossover

In general the crossover operator can improve global search capability for certain some bionic heuristics algorithms, and the design of crossover operator is important for bionic algorithms, especially when the algorithms are used to solving those high-dimensional, multi-modal functions. While any crossover operators are not employed in the basic ABC algorithm, so a novel, efficient crossover operator, name layer noise crossover (LNX) is introduced to the ABCP.

The fitness of every employed bees are not both equal, hence, we can sort the employed bee group according to their fitness. Then we divide the sorted population into three layers: the set A_{bt} saves the best 20% of population, the set C_{wt} save the worst 20% of population, and the remained individuals are saved in the set B_{md} . Next, three parents p_A, p_B and p_C are selected randomly in set A_{bt} , B_{md} and C_{wt} respectively. In the last step, the new individual is generated according to the formulation (3.3).

$$v = \alpha \times p_A + \beta \times p_B + \eta \times p_C \qquad (3.3)$$

where α, β and η are random numbers within $[0,1]$.

The LNX operation described in Eq.(3.3) is basically a discrete recombination[6]. Fig. 1 illustrates a two-dimensional and three- parent example, and the detailed description on LNX operator is list in table 3.

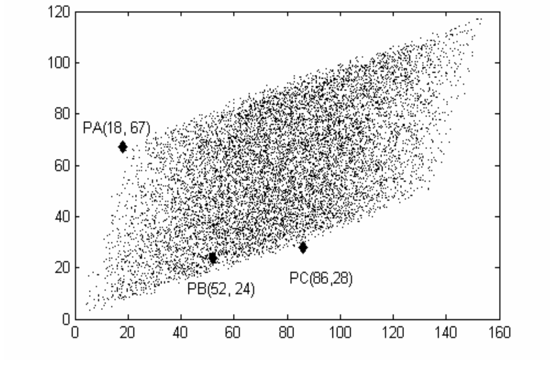

Figure.1. 10000 offspring are generated through three parents P_A, P_B and P_C .

Table 2. The Pseudo-code of LNX operator

```
1.  Sort EB according to its fitness;
2.  Divide EB into 3 sets: $A_{bt}$ , $B_{md}$ and $C_{wt}$
3.  For every $x \in A_{bt}$
4.      select $b \in B_{md}$ and $c \in C_{wt}$
5.      crossover base on Eq.(3.3).
6.  End
```

4. Experimental results

Experimental validation for the proposed ABCP is conducted on 15 classical benchmarks [7] function $f1$ - $f5$ are high dimensional, problems, $f1$ - $f5$ are unimodal functions. Function $f6$ is a step function which has one minimum and is discontinuous. Function $f7$ is a noisy quadratic function. Functions $f8 - f11$ are multimodal functions where the number of local minima increases exponentially with the problem dimension. Functions $f12 - f15$ are low-dimensional problems which only have one local minima[7].

The algorithms used for comparison are self-adaptive differential evolution (SaDE)[8] and basic ABC algorithm[2]. The experimental result of SaDE are adopted from ref.[8].

In the experiment, we set the maximum evolutionary generations as in ref [8]. The NP of ref[8] set to 100, while it is 15 in ABCP, then we set the FES to #gen*100 for fair comparison. The average results of 50 independent runs are listed in table 3.

The comparison show that ABCP gives better results on benchmark functions than the basic ABC algorithm, and ABCP performs better than SaDE on benchmark functions $f1, f2, f3, f7, f10, f11$, but SaDE gives better result than ABCP in function $f4$. On the remained benchmark functions, SaDE and ABCP both find same results.

Figure 2 shows average best fitness curve for the ABCP with 50 independent runs for selection benchmark functions $f1, f3, f11, f14$. In order to express more clearly the relationship between fitness and evolutionary generations, the fitness is transformed as Eq(4.1).

$$Fit = \log_{10}(fitness) \qquad (4.1)$$

5. Conclusion and discussion

In this paper we proposed a new ABC variant, artificial bee colony programming, an improved version of ABC algorithm. ABCP can be viewed as the hybridization of the basic ABC algorithm [2] and evolutionary programming [7]. In the paper we first analyze the principle of ABC algorithm in term of the selection scheme of onlooker bee group and the local and global search capabilities, then point out that its global search capability is weak. Next a random select scheme and LNX crossover operator were proposed. Finally, the ABCP performs better than SaDE and basic ABC algorithm on 15 benchmark functions.

Table 3. Experimental results, average over 50 independent runs of ABCP,ABC and SaDE algorithm. “Mean Best” indicates average of minimum value obtained, and “Std Dev” stands for standard deviation.

Function number	#Gen	EBC Mean Best (Std Dev)	ABC Mean Best(Std Dev)	SaDE Mean Best(Std Dev)
f_1	1500	8.283495×10^{-224}(0)	7.241185×10^{-5}(5.12×10^{-4})	1.1×10^{-28}(1.0×10^{-28})
f_2	2000	1.060799×10^{-192}(0)	0.200050(1.41)	1.0×10^{-23}(9.7×10^{-24})
f_3	5000	6.575917×10^{-197}(1.057584×10^{-90})	8.432284(29.90)	3.1×10^{-14}(5.9×10^{-14})
f_4	5000	1.774091×10^{-191}(0)	2.284732(0.99)	0(0)
f_5	20000	0(0)	0.065567(0.22)	0(0)
f_6	1500	0(0)	6.040000(13.22)	0(0)
f_7	3000	7.098318×10^{-5}(3.05×10^{-4})	0.031882(0.01)	3.15×10^{-3}(7.5×10^{-4})
f_8	9000	−12569.4866 (2.23×10^{-12})	−12426.5639 ($4.75\times10^{+2}$)	−12569.5(7.0×10^{-12})
f_9	5000	0(0)	1.749399(32.67)	0(0)
f_{10}	1500	8.881784×10^{-16}(0)	2.116504(1.48)	7.7×10^{-15}(1.4×10^{-15})
f_{11}	1500	1.570544×10^{-32} (0)	0.432317(0.89)	6.6×10^{-30}(7.9×10^{-30})
f_{12}	100	0.998003(0)	0.998004(9.17×10^{-15})	0.998004(2.4×10^{-16})
f_{13}	100	−1.031623 (4.06×10^{-5})	−1.031628(2.24×10^{-16})	−1.03163(9.7×10^{-12})
f_{14}	100	3.000005(1.78×10^{-11})	2.999999(2.41×10^{-15})	3.000005(1.7×10^{-15})
f_{15}	100	−10.402938(1.29×10^{-7})	−9.057995(2.55)	−10.4029 (4.9×10^{-7})

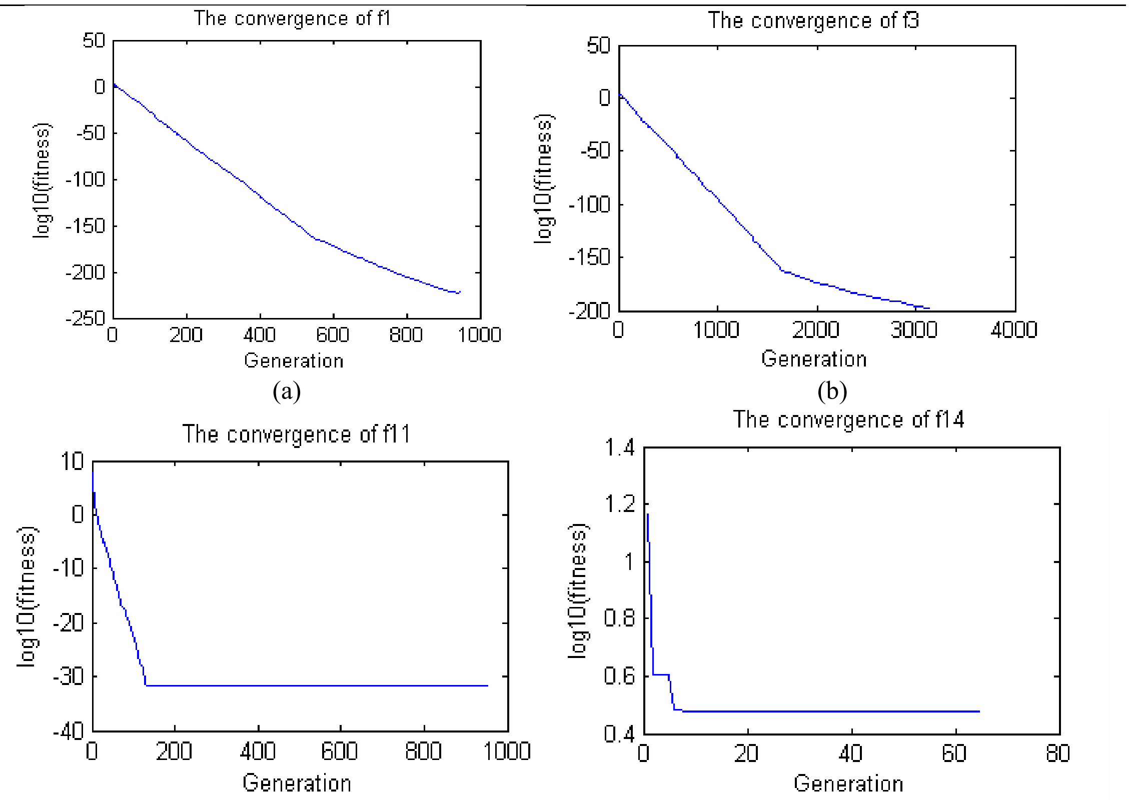

Figure.2. Average best fitness curves of ABCP for selected benchmark functions. All results are means of 50 runs. (a) Test function $f1$. (b) Test function $f3$. (c) Test function $f11$ (d) Test function $f14$

References

[1] D.Karaboga and D.Basturk. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony(ABC) algorithm". J Glob Optim, Vol. 39 2007, PP. 459–471.

[2] D.Karaboga and D.Basturk. "On the performance of artificial bee colony (ABC) algorithm". Applied Soft Computing, Vol. 8,2008, pp.687–697.

[3] V. Tereshko, T. Lee, "How information mapping patterns determine foraging behavior of a honey bee colony", Open Syst. Inf. Dyn. Vol. 9, 2002, pp.181–193.

[4] H.F. Wedde, M. Farooq, Y. Zhang. "BeeHive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior, ant colony, optimization and swarm intelligence", in: 4th International Workshop, ANTS 2004, Brussels, Belgium, 2004, pp.5–8.

[5] X.S. Yang, "Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms". Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach. 2005, pp. 317-323.

[6] R. Storn, K. V. Price, and J. Lampinen, Differential Evolution - A Practical Approach to Global Optimization, Springer, Berlin, 2005.

[7] X. Yao, Y. Liu, and G. Lin, "Evolutionary Programming Made Faster," IEEE Transaction on Evolutionary Computation, Vol.3,1999, pp.82-102.

[8] Brest J., Greiner S., Boskovic B., Mernik M., Zumer V.. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transaction on Evolutionary Computation, Vol. 10. 2006, pp. 646-657

.