# Research of Parallel Artificial Bee Colony Algorithm Based on MPI

Yingsen Hong
Department of Computer Science and Engineering
Harbin Institute of Technology
150001 Harbin, China
hongyingsen@139.com

Zhenzhou Ji
Department of Computer Science and Engineering
Harbin Institute of Technology
150001 Harbin, China
jzz@pact518.hit.edu.cn

Chunlei Liu
Department of Computer Science and Engineering
Harbin Institute of Technology
150001 Harbin, China
812liuchunlei@163.com

***Abstract*—Artificial bee colony algorithm is a smart optimization algorithm based on the bees acquisition model. A long time for the search of the artificial bee colony algorithm, in this paper we propose a parallel algorithm of artificial bee colony algorithm (MPI-ABC), with an application of a parallel programming environment MPI, using the programming mode of message passing rewriting the serial algorithm in parallel. Finally, this paper compare both serial and parallel algorithm with testing on complex function optimization problems. The experimental results show that the algorithm is effective to improve the search performance, especially for high-dimensional complex optimization problem.**

***Keywords-swarm intelligence theory; function optimization; artificial bee colony; parallel programming; MPI***

## I. INTRODUCTION

The artificial bee colony (Artificial Bee Colony, ABC) algorithm is based on the theory of swarm intelligence optimization algorithm. ABC algorithm in literature [1] proposed a swarm intelligence stochastic optimization algorithm in 2005, imitating bees swarm intelligence collecting nectar behavior. Bees carry out different activities in accordance with their respective division of labor, and achieve information sharing and exchange of bees to find the optimal solution of the problem. It has had a large number of applications in function optimization, combinatorial optimization, and engineering fields. But when facing the high-dimensional complex function, it exists searching a long time, slow in the convergence and prone to appear "premature" shortcomings. In this paper, we propose an MPI-based parallel artificial bee colony algorithm based on the ABC algorithm. Experiments of the parameter optimization problem show that MPI-ABC algorithm improves the search performance and it is very efficient and has a high practical value and potential applications.

## II. ARTIFICIAL BEE COLONY ALGORITHM

Karaboga put forward artificial bee colony algorithm based on Seeley bees self-organizing model in 2005, which is a simulation of the behavior of honey bees swarm intelligence optimization algorithm, achieving target search elements through the division of the swarm, providing with parameter settings simply, easy to implement and other characteristics.

Artificial bee colony algorithm consists of three parts composed of leading bees (employ bees), watching bees (onlooker bees) and investigation bees(scout bees), leading bees go to find food sources, the watching bees waiting for leading bees bring back food source in the dance area and making choices on food source based on information, the scouts completely randomly to find new food sources. If a food source is discarded by employ bees and scout bees, then the employ bee corresponded to the food source becomes a scout bee.

In algorithm, groups of solutions represent by SN D-dimensional vector, the first of $i$ solution can be expressed as $x_i$ , $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$, i=1,2,…,SN, food source corresponding to the amount of pollen solution quality (fitness value). The search process is as follows:

(1) Employ bees generate a new food source selection according to the formula (1):

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (1)$$

Among, $x_{ij}$ is a new location, and k is randomly selected, $k \in \{1,2,\ldots,BN\}$,(BN is the population number), moreover $k \neq i$, $j \in \{1,2,\ldots,D\}$,(D is the dimension of the target space), $\phi$ is a random number between[-1,1]. This step can also be suitably reduced. As the number of iterations cumulative, the distance between ( $x_{ij} - x_{kj}$ ) become narrow and the search space also become narrow, namely search step become narrow. Dynamically adjusting the step size is helpful to improve the algorithm accuracy and ultimately to obtain the optimal solution.

(2) Calculate the probability $P_i$ of employ bees to search for the quality of the food source, and onlooker bees select a food source according to the formula (2):

$$P_i = fit / \sum_{n=1}^{SN} fit_n \quad (2)$$

Of which, $fit$ is fitness value of the first of $i$ employ bee the corresponding food source.

(3) Onlooker bees generate new solution $v(i)$ according to the formula (2);

(4) For employ bees get into a local optimum, keeping limit(limit is an important control parameter in the ABC algorithm) bout iterations not change, and the fitness employ bees obtained is not the global optimum, abandon the food source, and replace with the food resource scout bees randomly searched. Let the solution be give up is $x_i$, then it is replaced by a new solution generated by scout bees. The updating formula is :

$$x_i = \min_j + rand(0,1)(\max_i - \min_j) \quad (3)$$

Of which, $j \in \{1,2,\cdots,D\}$, max and min denote respectively the maximum element and minimum elements in the collection $\{ x_1 , x_2 , \cdots , x_{SN} \}$after removing $x_i$ .

To sum up, onlooker bees transfer state on probability according to the size of the fitness, ensuring that most of the bees select a path in accordance with previous generations of historical information, and scout bees ensure that there is always a part of the bees randomly route, ensuring the diversity of solutions and this is helpful to escape from local optima. Otherwise, employ bees with elite characteristics retaining the previous generation of the optimal path can speed up the convergence of the algorithm and reduce the oscillation of the algorithm. It is the combined effect of the three to let the algorithm have stronger global search capability and the speed of convergence.

## III. PARALLEL ARTIFICIAL BEE COLONY ALGORITHM ANALYSIS AND DESIGN

### A. *The serial ABC algorithm performance analysis*

We use the Intel (R) Thread Profiler 3.1 Performance Analysis Tools to analysis the serial ABC algorithm in the 50 dimension, identify the number of processed block and rewritten in parallel. The results are shown in table I:

TABLE I. ABC ALGORITHM PERFORMANCE ANALYSIS TABLE

| Function | TotalTime(ms) | Callers | Callees | %in Function |
|---|---|---|---|---|
| CalculateFitness() | 57,525 | 3 | 0 | 100.00% |
| MemorizeBestSource() | 10,309 | 1 | 0 | 100.00% |
| Init() | 1,721 | 2 | 3 | 40.56% |
| Initial() | 1,049 | 1 | 1 | 1.33% |
| SendEmployedBees() | 657,811 | 1 | 3 | 46.24% |
| CalculateProbabilitiies() | 12,039 | 1 | 0 | 100.00% |
| SendOnlookerBees() | 852,150 | 1 | 3 | 44.70% |
| SendScoutBees() | 6,268 | 1 | 1 | 89.06% |
| main() | 4,808,938 | 1 | 8 | 67.98% |

### B. *Parallel design of the bee colony algorithm*

From Table I, we can know, throughout the algorithm, function main, SendEmployedBees and SendOnlookerBees the three functions consuming the longest. We make parallel design for these three functions. We set that new algorithm runs on two nodes, n processes, each process computes respectively and statutes to the main process in final.

There are two load program in MPI, static load program and dynamic load program. Static load balancing is a basic method to achieve load balancing of MPI parallel programs. It means how to assign the task before the MPI parallel programs run. That expects each process to try to complete their tasks at the same time in order to effectively reduce the program running time. Generally speaking, each node running a process, the static load balancing is how to assign tasks on each node.

Dynamic load balancing is an effective complement to static load balancing, referring that during the MPI parallel program running, according to each computing node in the current load, dynamic migration of tasks between the compute nodes, migrating the task on the heavy load of the node to the light load node, trying to balance the load of the compute nodes, thus reducing program run time. When estimating inaccurately of the assignments, dynamic load balancing can obtain excellent results. Compared to static load balancing, dynamic load balancing does not need to know the amount of task and the computing power of the node before MPI parallel program run tasks. But dynamic load balancing is more complex in implement, it needs some additional communication cost.

This paper uses the static load program. First, we calculate the amount of assigned tasks required on each node, for the amount of assigned tasks on each node is proportional thereto the computing capacity of the node. So that each node expects to complete the task at the same time, thereby minimizing the program run time.

MPI-ABC algorithm main steps are as follows:

Step1: Initialize the MPI runtime environment;

Step2: If the current process is the main process, get algorithms related to the initial parameters such as population size SN. And sent the above-mentioned parameters and split the sample to each slave processes. Each process randomly generate the initial nectar location constituted of SN solutions.

Step3: Employ bees of each process in accordance with the formula (1) search for new nectar, and calculate the position and moderation. If the new location is better than the original location, then replace the original location.

Step4: Scout bees of each process select a nectar location depend on probability according to the nectar amount of bees sources, and generate a new location according to the formula (1). After evaluating of the location, if the new location is better than original location, replace the original location.

Step5: Each process finds the existence of abandoned nectar, where the employ bees become scout bees, and where the nectar is replaced of random nectar.

Step6: Each process compares the number of iterations cycle with maxcycle, if (cycle > maxcycle) then records the optimal solution, otherwise goes to Step3.

Step7: Each process statutes the results from the slave process to the main process. Exit parallel and output a final best solution.

## IV. EXPERIMENTS AND ANALYSIS

### A. *Experimental platforms and parameters*

The experimental hardware environment is Intel(R) Core(TM) 2 Quad Dual-core PC, memory 2 G. Software environment is Linux 2.6.23.1-42.fc8. Performance testing tool is Intel (R) Thread Profiler 3.1 Performance Analysis Tool. In the experiment, SN=20, BN=SN/2, we take D = 50 dimension and D = 100 dimension, maxcycle=2500, limit=100. Test function independently executes 30 times to find the average. The experiment selects three benchmark functions to carry out comparison test from references [6].

(1) Sphere function:

$$f(x)=\sum_{i=1}^{n} x_i^2 \quad -100 \le x_i \le 100$$

It is a continuous unimodal function, when $x_i$ = 0 ($i$ = 1,2, ..., n) the function reaches the minimum value 0.

(2) Griewank function:

$$f(x)=\frac{1}{400}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1 \quad -600\le x_i \le 600$$

It is a complex nonlinear multi-peak function, when $x_i$ = 0 ($i$ = 1,2, ..., n) the function reaches the minimum value 0.

(3) Rastrigin function:

$$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10) \quad -5.12\le x_i \le 5.12$$

It is a complex nonlinear multi-peak function, when $x_i$ = 0 ($i$ = 1,2, ..., n) the function reaches the minimum value 0.

### B. *Experimental Analysis*

Analysis I: In 100 dimension, as can be seen from Table II that MPI-ABC algorithm is better than ABC algorithm in aspect of Calculating the Maximum and Mean of the function, this fully shows that the improved algorithm has good global search performance.

TABLE II. COMPARISON OF TEST FUNCTION'S OPTIMIZATION RESULTS (D=100)

| Function Name | Algorithm | Maximum | Mean |
|---|---|---|---|
| Sphere | ***ABC*** | 2.887775e-05 | 2.934553e-06 |
| | ***MPI-ABC*** | 6.462004e-05 | 6.084885e-06 |
| Griewank | ***ABC*** | 6.904620e-07 | 2.625769e-08 |
| | ***MPI-ABC*** | 2.405992e-02 | 4.278692e-03 |
| Rastrigin | ***ABC*** | 2.173169e+01 | 1.146211e+01 |
| | ***MPI-ABC*** | 1.835299e+01 | 1.226773e+01 |

Analysis II: If each ABC algorithm average running time is $T_S$ and each MPI-ABC algorithm average running time is $T_P$ ,then the speedup ratio is $S_P$ :

$$S_p = T_s / T_p$$

Table III and Table IV are the speedup ratio of two algorithms in 50 dimension and 100 dimension, and it can be concluded that using MPI parallel algorithm has effectively improved the computing performance and speedup ratio performance. Among of the 100 dimension Griewank function has improved more than 50%, the other two function also increased by nearly 50%. This illustrates that MPI-ABC algorithm has its superiority in solving high-dimensional global search space problem.

TABLE III. OPERATING EFFICIENCY OF THE ALGORITHM IN 50 DIMENSION

| Function Name | Runging Time (D=50) | | Speedup Ratio |
|---|---|---|---|
| | ***ABC(s)*** | ***MPI-ABC(s)*** | |
| Sphere | 0.8700 | 0.6525 | 1.3333 |
| Griewank | 5.7200 | 3.7423 | 1.5285 |
| Rastrigin | 4.6100 | 2.4913 | 1.8504 |

TABLE IV. OPERATING EFFICIENCY OF THE ALGORITHM IN 100 DIMENSION

| Function Name | Running Time(D=100) | | Speedup Ratio |
|---|---|---|---|
| | ***ABC(s)*** | ***MPI-ABC(s)*** | |
| Sphere | 1.7900 | 0.9074 | 1.9723 |
| Griewank | 14.5600 | 7.1085 | 2.0483 |
| Rastrigin | 9.2300 | 5.0940 | 1.8119 |

Analysis III: As shown in Figure 1, Figure 2 and Figure 3 are respectively Sphere funtion、Griewank function and Rastrigin function in 1-500 dimension, the running time of ABC function and MPI-ABC function. Among of these MPI-ABC runs on 2 nodes. From the figures, we can see that speedup ratio of three functions is 2, this also explains the superiority of MPI-ABC algorithm for solving large search space of high-dimensional global problem.

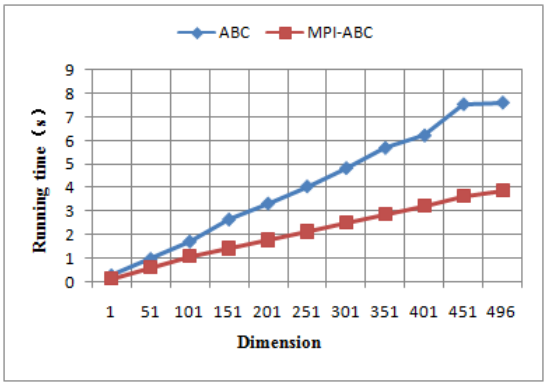


Figure 1. Sphere function's running time in some dimension.

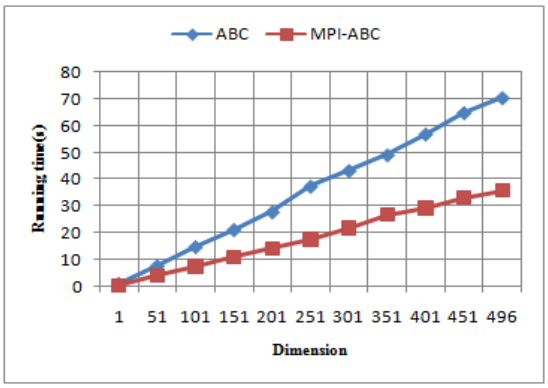


Figure 2. Griewank function's running time in some dimension

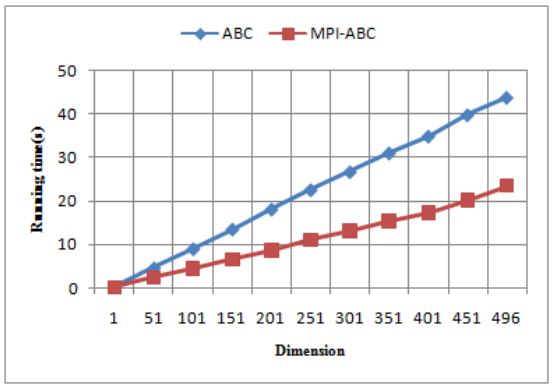


Figure 3. Rastrigin function's running time in some dimension

## V. CONCLUSION

In this paper after studying the traditional artificial bee colony algorithm, on the basis of it, using MPI parallel programming model, we study and design artificial bee colony MPI parallel algorithms under clusters. The experiment result show that the algorithm in high dimension, has a high operating efficiency and provide an effective means for the high-dimensional complex optimization problem. For the case of low dimension, due to corresponding parallelism overhead of process creation and destruction overhead, the speed of parallel processing and serial processing is little different. Overall, Using MPI parallel programming model has been greatly improved the entire performance of the ABC algorithm.

## ACKNOWLEDGMENT

This research was supported by the National Natural Science Foundation of China under Grant No. 61173024.

## REFERENCES

[1] Karaboga D. An idea based on bee swarm for numeri-cal optimization,TR06[R].[S.l.]: Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[2] LIU Hao, YANG Hui, YIN Zhong-ke, et al.Signal sparse decomposition based on MPI parallel computing[J]. ComputerEngineering, 2008, 34(12): 9-21.

[3] CHOU Y C, Nestinger S S, Cheng H H.Ch MPI : Interpretive parallel computing in C[J]. Computing in Science and Engineering, 2010,12(2): 54-67.

[4] CHEN Xing, HUANG Ka-ma. Constructing a Beowulf parallel computing system based on Windows and MPI[J]. Computer Engineering and Applications, 2003, 39(4): 59-61.

[5] LI Shuang, LI Wen-jing, YANG Wen, ZHOU Hai-yan. Research and Implementation of Parallel Random Perturbation Artificial Bee Colony Algorithm Based on Multi-core PC[J]. MICROELECTRONICS & COMPUTER, 2012, 29(9): 63-66.

[6] Karaboga D, Basurk B. On the performance of Artificial Bee Colony(ABC) algorithm[J]. Applied Soft Computing, 2008, 8(1): 687-697.

[7] GAO Wei-feng, LIU San-yang, JIANG Fei, ZHANG Jian-ke. Hybrid artificial bee colony algorithm[J]. Systems Engineering and Electronics, 2011, 33(5): 1167-1170.

[8] LU Yun-e, HUANG Zong-yu, LI Chao-yang, GUO Xiang-bin, YIN Hui-ming. The MPI parallel computing based on the microcomputer cluster[J]. Electronic Design Engineering, 2011, 19(5): 78-81.

[9] LU Ke-zhong, LIN Xiao-hui. Implementing Load Balance in MPI Parallel Program[J]. Microcomputer Information, 2007, 23(5-3): 226-227.