An Improved Artificial Bee Colony Algorithm for the Task Assignment in Heterogeneous Multicore Architectures

Tao Zhang[1,2], Xuan Li[1,2(✉)], and Ganjun Liu[1,2]

[1] School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China
lixuantju@tju.edu.cn

[2] Texas Instruments DSP Joint Lab, Tianjin University, Tianjin 300072, China

Abstract. The Artificial Bee Colony (ABC) algorithm is a new kind of intelligent optimization algorithm. Due to the advantages of few control parameters, computed conveniently and carried out easily, ABC algorithm has been applied to solve many practical optimization problems. But the algorithm also has some disadvantages, such as low precision, slow convergence, poor local search ability. In view of this, this article proposed an improved method based on adaptive neighborhood search and the improved algorithm is applied to the task assignment in Heterogeneous Multicore Architectures. In the experiments, although the numbers of iteration decreases from 1000 to 900, the quality of solution has been improved obviously, and the times of expenditure is reduced. Therefore, the improved ABC algorithm is better than the original ABC algorithm in optimization capability and search speed, which can improve the efficiency of heterogeneous multicore architectures.

Keywords: Artificial bee colony algorithm · Task assignment
Neighborhood search

1 Introduction

Nowadays single core architectures is gradually replaced by multiple cores due to problems in obtaining further performance increases from single core processors [1]. Heterogeneous multicore architecture (HMA) is an integration of special purpose processing cores. The purpose of task assignment in HMA is to help system designers to get the best-performance and the lowest-cost design scheme in HMA [2]. Task assignment in HMA minimizes the execution time and consumed power of the target system [3] with certain constraint conditions.

Task assignment in HMA is an NP-hard problem [4]. Many heuristic algorithms have been applied to solve the NP-hard problem [5–8]. Due to the advantages of few control parameters, computed conveniently and carried out easily, ABC algorithm has been applied to solve many practical optimization problems [9], which have obtained preferable results.

ABC algorithm is based on swarm behavior, with the characteristics of integrity, relevance, dynamic and orderliness on systematics [10]. It can be evolved from disorder to order by self-organizing. Bees and bee colonies have feedback features at the same time [11]. Because it is limited by the way of evolution, there are still some disadvantages, such as low precision, slow convergence [12], poor local search ability [13–15]. In this paper, we describe the HMA as a model of a task assignment. The HMA are combined with two different cores. The original ABC algorithm is used in the model to achieve the DAG diagram corresponding to system tasks. Then an improved method based on adaptive neighborhood search is proposed to address the original ABC algorithm's disadvantages. Five DAG figures are generated randomly with TGFF [16] tools as a test set in order to compare the performance of the original and the improved ABC algorithms. The experimental results demonstrate that the improved ABC algorithm is more efficient.

2 Task Assignment Based on ABC

2.1 Bees Behavior

ABC simulates the co-operation, mutual coordination between individuals and groups in intelligent foraging and breeding behavior of bee swarms. They exchange information through dance and odor to finish foraging behavior. In a typical ABC algorithm, three types of artificial bees are considered as agents for solving an optimization problem, called employed bees, onlooker bees and scout bees. They have their own division or labor in foraging. The scout bees are responsible for investigation. The employed bees and onlooker bees are responsible for the exploitation of food sources. The bees maintained good coordination to achieve a better balance, and then completed the bee groups of foraging, reproduction and other behaviors.

2.2 Mathematical Model of the Artificial Bee Colony Algorithm

A system task is divided into a number of sub-tasks which can be completed by a combination of core A (represented by 0) and core B (represented by 1). When the two cores process the same task they consume different time and power. The coded information corresponding to the task assignment can be seen as an ordered set of binary numbers. The task assignment in HMA can be abstracted as a multi-objective combinatorial optimization problem in mathematics, to find a sub-optimal solution or the optimal solution under different constraints.

Bees can quickly find a better food source in foraging. Similarly, the task assignment can quickly find a better solution in the process. The correspondence among bee foraging behavior, mathematical model and task assignment is shown in Table 1.

Apparently, the more nodes there are, the more task assignment schemes. The number of schemes is growing at an exponential rate to the number of nodes. In a DAG, some tasks are completed by core A, the others are completed by core B. Figure 1 presents one scheme of the task assignments.

The best scheme from all the task assignment schemes is to be selected according to their fitness values and the probability to be searched. The maximum fitness food source is selected, which may be the optimal or suboptimal solution corresponding to

Table 1. The correspondence among Bee foraging behavior, mathematical model and task assignment

Bee foraging behavior	Mathematical model	Task assignment
All the food sources	All the solutions	All the Task assignment
The location of a food source	One solution	One Task assignment scheme
The best food source	The best solution	The best Task assignment scheme
The fitness of a food source	The quality of a solution	Performance of the system
Bees' foraging speed	Solutions' convergence rate	The speed of optimization capability

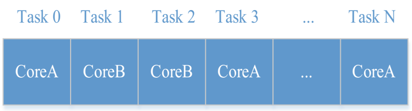

Fig. 1. One scheme of the task assignments.

the best task assignment in HMA. The Eq. (1) is used to describe the best scheme of task assignment in HMA.

$$\begin{cases} Max\_fitness := \max(fitness[i]) \\ \qquad \sum_{j=0}^{N-1} T_j < Time\_Limit \\ s.t. \\ \quad \sum_{j=0}^{N-1} Power_j < Power\_Limit \end{cases} \tag{1}$$

Here $fitness(i)$ is the fitness value of i-th food source, T_j and $Power_j$ are the total time consuming and the total power consuming of the j-th node respectively. $Time\_Limit$ and $Power\_Limit$ are the maximum time and the maximum power limitations respectively.

2.3 Description of the Original ABC Algorithm

ABC is applied to solve the traveling salesman problem in literature [11]. The algorithm based on the mathematical model in Eq. (1) is applied to task assignment in HMA.

At first, the bees are equally divided into scout bees and onlooker bees. The scout bees search the food source and the food source in its neighborhood. If the fitness of its neighborhood position is greater than its fitness, then the location of i-th food source is substituted by the location of its neighborhood. Onlooker bees are sent according to the probability of each candidate food source to search food. The greater the probability is, the greater likelihood of neighborhood search is. If the fitness of the neighborhood position is greater than that of the current location, the current location is substituted by the location of its neighborhood.

When the time consumed by the neighborhood search is greater than the maximum limit time of food source, the food source is initialized. When its search time is greater than the global maximum limit, the worst food source is initialized. Every bit of coding information (0 or 1) is selected probably in the initialization.

At last the optimal or suboptimal food source searched is recorded. The information coding corresponds to the best scheme of task assignment in HMA. In original ABC algorithm, scout bees and onlooker bees finish their neighborhood search by updating one bit of information coding randomly, such as 0 (1) is updated as 1 (0). So the information update of food source is implemented.

3 The Analysis and Improvement of ABC Algorithm

3.1 The Shortcoming of the Original ABC Algorithm

In ABC algorithm, food source is updated through replacing the sub-optimal position by optimal position. The food-searching process is equivalent to the process of finding the optimal solution to task assignment in HMA.

When the initial position of the food source is far away from the optimal food source, there is a big difference between the encoded information of them. The method mentioned above will be very low efficiency. Obviously, the convergence rate will be lower and the method will increase the number of invalid iterative search. It is a broader range and high-discrete solution space. Finding the optimal solution is a continuous iteration and selection process. Therefore, in order to improve the optimization capability and reduce the time overhead, invalid iterations should be avoided or minimized.

3.2 The Improvement of ABC Algorithm

In view of the deficiencies of the original ABC algorithm, the efficiency of ABC algorithm is improved from the aspects of invalid iteration and neighborhood search strategy.

The Improvement of Neighborhood Search Strategy
When neighborhood search iteration is less than the constraint condition, the neighborhood search strategy updates one bit of the coded information in an iteration. On the contrary, when neighborhood search iterations exceed the constraint condition, but the fitness of the corresponding food source cannot be improved, then, the neighborhood search strategy is changed to update several bits of the coded information in an iteration.

Function named Search_Neighbourfood_Strategy_Improved([i]) is used to updates the coded information of the food source randomly. The pseudo-code is as follows (Table 2):

Table 2. The pseudo-code of the new neighborhood search program for the *i*-th food source.

```
Function: Search_Neighbourfood_Strategy_Improved([i])
begin
        /*update several bits of the coded information to realize the neighborhood search*/
        $X_{i,neighbour\_location:=Update\_nbits}(X_{i,old\_location})$;
1       /*The location of neighborhood food source is assigned to the current food source*/
        $X_{i,old\_location} := X_{i,neighbour\_location}$;
2   end
```

Here, $X_{i,old\_location}$ is the information coding of the i-th food's current location, $X_{i,neighbour\_location}$ is the information coding of the i-th food source's neighborhood.

The Improvement of Calculating the Worst Food Source
During later iteration, all of the current food sources are approaching the final result. The worst food source is re-coded according the best food source's current location instead of initialize the worst food source. Several bits of the best food source's current location are kept and the remaining bits are re-coded. The pseudo-code is as follows (Table 3):

Table 3. The pseudo-code of generating a new solution.

```
Function: SendScoutBees_new([i])
begin
        /*update several bits of the coded information to realize the neighborhood search*/
        $X_{neighbour\_location} := update\_nbits(X_{best\_location})$;
1       /*The location of neighborhood food source is assigned to the new food source*/
        $X_{new\_location} := X_{neighbour\_location}$;
2   end
```

Here, $X_{best\_location}$ is the information coding of the current best food source's location, $X_{neighbour\_location}$ is the information coding of the current best food source's neighborhood, $X_{new\_location}$ is the information of the new solution.

4 Experimental Results and Analysis

Algorithms in this paper are implemented in C-language and tested on a computer with Intel core i5-4460 and 8 GB RAM. The running environments consists of Windows 10 and Microsoft Visual Studio 2013 Ultimate. In the experiment we set population size as 15. Food sources number is equal to 0.5 * population size. The number of initial time scout bees is equal to the follow bee, which scale is equal to 0.5 * population size.

Five DAGs generated randomly with TGFF tool are regarded as a sample set to compare the test results of the original ABC algorithm and the improved ABC (I-ABC) algorithms. The parameter settings are shown in Table 4.

Table 4. Parameter settings

Parameters	Values
CoreA execution time	[392 ms, 450 ms]
CoreA consumed power	[25 w, 35 w]
CoreB execution time	[72 ms, 84 ms]
CoreB consumed power	[264 w, 336 w]

The comparison results between the ABC and the I-ABC algorithms are shown in Table 5.

Table 5. Comparison results between the ABC and the I-ABC algorithms

Nodes	Algorithm	Iterations	Optimal solution	The worst solution	Average solution	Average consumed power	Average execution time
30	ABC	1000	1590.1074	1911.0752	**1801.8261**	5970.2704	0.2317 s
30	I-ABC	900	1238.3132	1163.5138	**1418.6363**	5962.4877	0.2214 s
52	ABC	1000	1618.9009	2471.8289	**2116.6482**	1053.5813	1.5177 s
52	I-ABC	900	1493.8208	2072.0464	**1629.4947**	1055.6977	1.4246 s
75	ABC	1000	1791.9229	2784.7266	**2337.8188**	1544.2631	6.1917 s
75	I-ABC	900	1660.3372	2356.6123	**1914.4037**	1551.2477	5.8043 s
102	ABC	1000	2104.1284	3092.9399	**2600.3895**	2066.0012	13.6863 s
102	I-ABC	900	1968.8102	2662.5608	**2221.5824**	2090.9569	12.9610 s
132	ABC	1000	2708.5594	3218.9221	**2934.5233**	2905.2343	34.2272 s
132	I-ABC	900	2701.4824	3028.0132	**2835.6697**	2998.8134	32.6838 s

To illustrate the effectiveness of the I-ABC algorithm, its performances are shown in Figs. 2 and 3.

According to the comparison in Table 5, Figs. 2 and 3, data could be analyzed from the following three aspects:

The Local Search Ability: Although the number of iterations is reduced from 1000 to 900, the optimal solution of the improved algorithm is not only not worse, even better than that of the original algorithm. It is more obvious for the case of 52 nodes. From 30 nodes to 132 nodes, the quality of the solution increases about the 21.27%, 23.02%, 18.11%, 14.57%, 3.37% respectively. Meanwhile the time-consumed is reduced, and the power-consumed is the similar. Therefore, the I-ABC algorithm shows better local search ability.

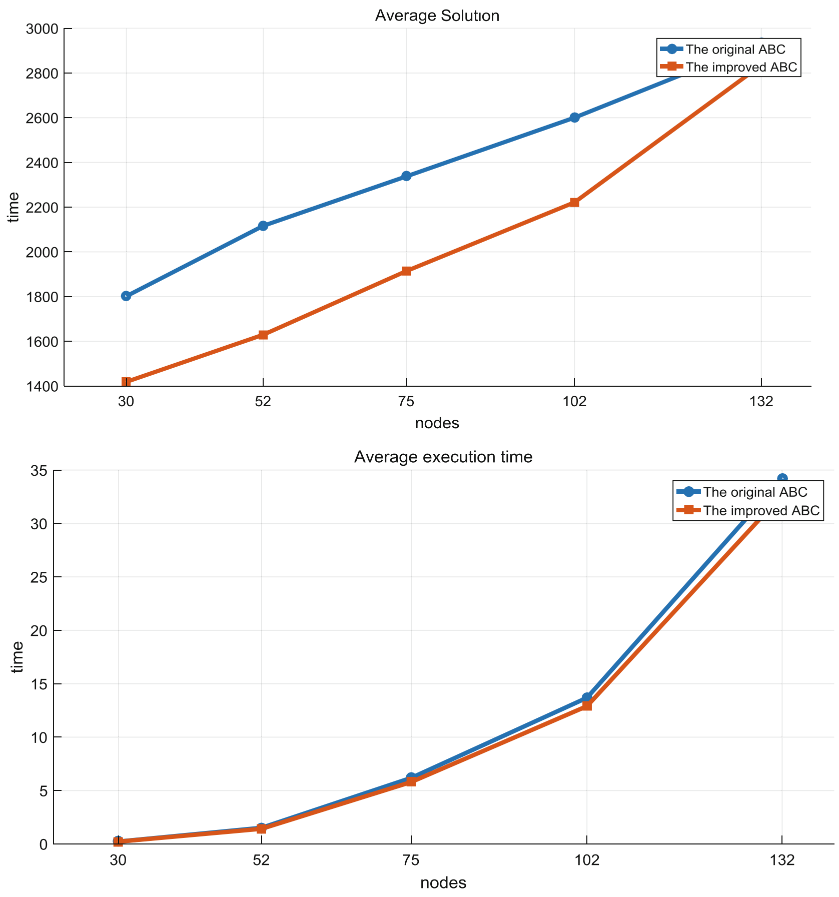

Fig. 2. Comparison of two algorithms over the average solution and the average execution time.

The Solution Accuracy: The D-value between the worst solution and the optimal solution in the I-ABC algorithm is reduced obviously. The final solution is also reduced significantly. For the case of 132 nodes, the final solution reduced about 3.5%, so the algorithm's accuracy is improved.

The Convergence Speed: When the consumed power is substantially the same, the improved algorithm can reduce the average execution time of the task.

From what we have been discussed above, we can get a conclusion that the precision of optimal solution of the I-ABC algorithm is higher than that of the original ABC algorithm, and the average execution time is reduced. The I-ABC algorithm is more efficient.

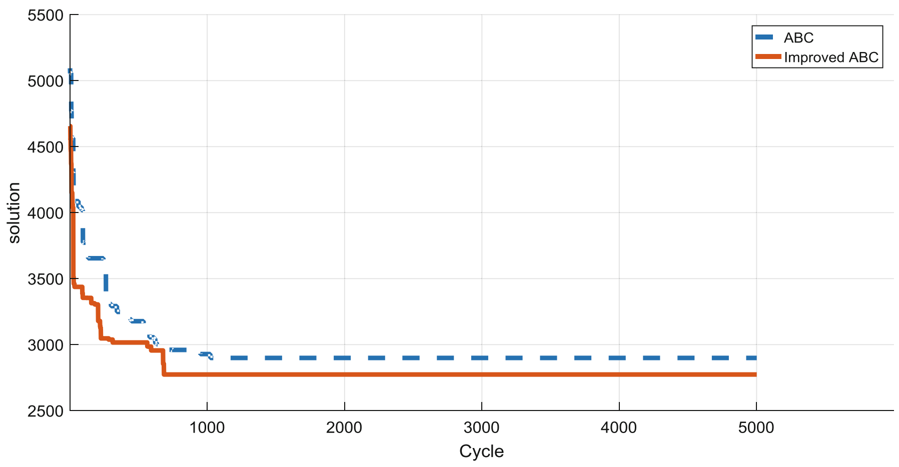

Fig. 3. Comparison results between the ABC and the I-ABC algorithms (132 nodes).

5 Conclusion

ABC algorithm has the features of less parameter and strong robustness, for which it is used in task assignment in HMA. Based on the original ABC algorithm, the neighborhood search scheme is proposed to solve the problems of low precision, slow convergence and poor local search ability. The experimental result shows that, in the case of less iterations, the I-ABC algorithm reduces the task execution time, and in case that the power-consumed obeys the restrictive conditions, the I-ABC algorithm reduces average solution with reduced numbers of iteration. D-value between the worst solution and the optimal solution is also reduced. The task assignment in HMA could be more efficient with the I-ABC algorithm.

References

1. Ya-Shu, C., Chiang Liao, H., Ting-Hao, T.: Online real-time task scheduling in heterogeneous multicore system-on-a-chip. IEEE Trans. Parallel Distrib. Syst. **24**(1), 118–130 (2013)
2. Hayashi, A., Wada, Y., Watanabe, T., Sekiguchi, T., Mase, M., Shirako, J., Kimura, K., Kasahara, H.: Parallelizing compiler framework and API for power reduction and software productivity of real-time heterogeneous multicores. In: Cooper, K., Mellor-Crummey, J., Sarkar, V. (eds.) LCPC 2010. LNCS, vol. 6548, pp. 184–198. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19595-2_13
3. Fred, A.B., Daniel, J.S., Landon, P.C.: The impact of dynamically heterogeneous multicore processors on thread scheduling. IEEE Micro **28**(3), 17–25 (2018)
4. Jing, L., Kenli, L., Dakai, Z., et al.: Minimizing cost of scheduling tasks on heterogeneous multicore embedded systems. ACM Trans. Embed. Comput. Syst. **16**(2), 1–25 (2016)
5. Lanying, L., Yan-bo, S.: New Genetic algorithm and simulated annealing integration of Hardware/Software partitioning. Comput. Eng. Appl. **46**(28), 73–76 (2010)

6. Jianliang, Y., Manmam, P.: Hardware/Software partitioning algorithm based on wavelet mutation binary particle swarm optimization. In: 3rd International Conference on Communication Software and Networks, pp. 347–359. IEEE (2011)
7. Ahmed, U., Khan, G.N.: Embedded system partitioning with flexible granularity by using a variant of tabu search. In: Canadian Conference on Electrical and Computer Engineering, pp. 2073–2076. IEEE (2004)
8. Hai, Y., Xiao-ya, F., Sheng-bing, Z., et al.: A guiding function based greedy partitioning algorithm for dynamically reconfigurable systems. In: 8th International Conference on Solid-State and Integrated Circuit Technology, pp. 2009–2012. IEEE (2007)
9. Dengxu, H., Ruimin, J., Shaotang, S.: An article bee colony optimization algorithm guided complex method. In: 5th International Symposium on Computational Intelligence and Design, pp. 348–351. IEEE (2012)
10. Wei, Z., Jing, L., Jian-chao, Z.: Artificial bee colony algorithm and its application in combinatorial optimization. J. Taiyuan Univ. Sci. Technol. **1**, 108–112 (2010)
11. Li, L., Cheng, Y., Tan, L., Niu, B.: A discrete artificial bee colony algorithm for tsp problem. In: Huang, D.-S., Gan, Y., Premaratne, P., Han, K. (eds.) ICIC 2011. LNCS, vol. 6840, pp. 566–573. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24553-4_75
12. Jun, L., Qian, W.: A modified artificial bee colony algorithm based on converge-onlookers approach for global optimization, pp. 10253–10262. Applied Mathematics & Computation, 219(20) (2010)
13. Guopu, Z., Sam, K.: Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. **217**(7), 3166–3173 (2010)
14. Wang, H., Liu, J., Wang, Q.: Modified artificial bee colony algorithm for numerical function optimization. Comput. Eng. Appl. **48**(19), 36–39 (2012)
15. Bai, L., Li-gang, G., Wen-lun, Y.: An improved artificial bee colony algorithm based on balance-evolution strategy for unmanned combat aerial vehicle path planning. Sci. World J. **2014**(1), 95–104 (2014)
16. Dick, R.P., Rhodes, D.L., Wolf, W.: TGFF: task graphs for free. In: Proceedings of the Sixth International Workshop on Hardware/Software Codesign, (CODES/CASHE 1998), pp. 97–101. IEEE (1998)