International Journal of Control and Automation
Vol.7, No.9 (2014), pp.345-356
http://dx.doi.org/10.14257/ijca.2014.7.9.30

Scheduling Tasks on Heterogeneous Multi-Core Processors Based on Modified Ant Colony Optimization

Zhulin Li[1,2*],Cuirong Wang[3], Haiyan Lv[4] and Xin Song5

[1]*College of information science and engineering,Northeastern University, Shenyang, Liaoning, China*
[2]*Modern educational technology center, Shenyang Agricultural University, Shenyang, Liaoning, China*
[3]*College of computer and communication engineering,Northeastern University at Qinhuangdao, Qinhuangdao, Hebei, China*
[4]*College of forestry, Shenyang Agricultural University, Shenyang, Liaoning, China*
[5]*Computer center, Northeastern University at Qinhuangdao,Qinhuangdao, Hebei, China*
Corresponding author's email: zhulin@syau.edu.cn

Abstract

Traditional techniques of single CPU no longer fit the trend of big data processing and parallel computing environment. Accordingly, heterogeneous multi-core processors bring opportunities. However, task scheduling strategies on heterogeneous multi-core processors are not as mature as the hardware techniques. To this end, in this paper, we study on the task scheduling algorithm on heterogeneous multi-core processors. Specifically, we assume there exists dependence between tasks, and due to the different computing capabilities of processors, the workload on each processor should be considered as well. Therefore, we formulate the problem as to optimize the makespan and load balance of the whole task execution process, and propose a modified Ant Colony Optimization (ACO) algorithm to find the optimal solution. Experiments are also conducted for evaluation.

Keywords: *Task Scheduling, Multi-Core Processors,Ant Colony Optimization*

1. Introduction

With the increasing development of information techniques, applications of big data processing [1] and parallel computing [2] presents more requirements on the performance of processors. However, with the restrictions of integrated circuit techniques of single CPU, it is hard to improve the performance by increasing the frequency only. Accordingly, the development of multi-core processors [3-4] conforms to the trend.

However, as indicated by Amdahl law [5], there still remains problem in improving performance through increasing the number of processors on a single chip, since the overall system would be restricted to the sequential execution part of the computing. Besides, adding more cores to the same chip brings more energy burdens [6]. To this end, heterogeneous multi-core processors with different computing capabilities and structures can accelerate the execution of tasks by assigning them to appropriate processors.

Even though the hardware technique of processors architecture is mature, the task scheduling strategies on heterogeneous multi-core processors still need corresponding revolution. Therefore, in this paper, we study on the task scheduling problem on heterogeneous multi-core processors. Specifically, we assume there exists dependence between tasks, and the execution sequence influences the overall performance. Besides, due to the different computing capabilities of processors, the workload on each processor

ISSN: 2005-4297 IJCA
Copyright ⓒ 2014 SERSC

would be different as well. In order to achieve the best efficiency of the system, we also consider the load balance factor. Therefore, we try to optimize the makespan and load balance of the whole task execution process by scheduling tasks to proper multi-core processors.

Ant colony optimization (ACO) [7] is a swarm intelligence method to find approximate solutions to difficult optimization problems. The basic idea is to find the shortest path by the amount of pheromones released by ants on each trail. It has been proved that employing ACO for task scheduling problems is feasible [8-10]. In this paper, we modified ACO algorithm to schedule tasks on heterogeneous multi-core processors with the objective to optimize the makespan and load balance of the whole task execution.

The remain of this paper is organized as follows. Section 2 discusses the related work, and Section 3 describes the problem statement. In Section 4, the proposed task scheduling algorithm based on ACO is presented, and in Section 5 we conduct some experiments for evaluation. Finally, the paper is concluded in Section 6.

2. Related Work

Generally, related work can be categorized into two groups: task scheduling algorithms and efforts on ant colony algorithms.

Many efforts have been made on task scheduling, including clustering based [11], list based [12] and replication based [13] methods. For example, Yang *et al.*, [14] proposed Dominant Sequence Clustering (DSC) algorithm keeps track of the critical path of the task graph and adjusts the critical nodes into one cluster. Critical Path Fast Duplication (CPFD) classifies three kinds of nodes with priorities: Critical Path Node (CPN), In-Branch Node (IBN) and Out-Branch Node (OBN), and then do the scheduling process. Kruatrachue *et al.*, Proposed the Duplication Scheduling Heuristic (DSH) [15] algorithm with the consideration of intervals between the task execution and replication.

There are plenty of works on ant colony algorithms. For example, Seo *et al.*, [16] employed ACO to solve the job scheduling problem in workshops. Zhang et al. [17] proposed an adaptive ACO to solve TSP problem, using evolution coefficient to evaluate the solutions of different ants. Ellabib *et al.*, [18] designed a multi-ant system to solve the vehicle routing problem with time window. Yang *et al.*, [19] employed multi-ant systems to solve the data clustering problem, where within each cluster, ants have different pheromones and transfer probabilities. Ghoseiri *et al.*, [19] developed a multi-object ACO algorithm to solve the bi-objective optimization problem. Different from existing works, in this paper, we introduce idea from immune algorithm to improve the diversity of solutions and accelerate the search.

Besides, there are also some efforts on applying swarm intelligence algorithms to solve task scheduling problem in cloud computing environment. For example, [20] proposed to use genetic algorithm to schedule independent tasks in cloud computing environment. [21] designed a PSO based algorithm to optimize the communication and execution time. [22] tried to minimize the makespan using ACO algorithm. Different from above works, we aim to optimize the makespan as well as the load balance of the whole system with the improved ACO algorithm over the local optimal and low convergence phenomenons.

3. Problem Statement

Generally, the task allocation problem can be described as follows: we aim to find an optimal solution to assign n tasks onto m resources, in order to improve the overall performance. The metrics of evaluating the performance of the system include optimal makespan, Quality of Service (QoS), load balancing and economic principles [6]. In this paper, we consider the objective is to minimize the total makespan and also achieve load balance. The reason behind is the computing capability of resources in a heterogeneous

 Copyright ⓒ 2014 SERSC

multi-core system varies, and the more balance the overall workload is, the more efficient the execution is.

Specifically, we assume the scope of our task scheduling problem is as follows. (1) The multi-core processors are heterogeneous, which means the computing capability varies for different processors and the scheduling method should consider the execution performance on each core. (2) We study on static scheduling only. That is, the number of tasks, communication cost between them, and the dependence relationship between data remain unchanged during the execution, and the assignment and execution sequence are deterministic. (3) There would be dependence and communication cost between tasks, and therefore the sequence of tasks execution as well as their time costs are significant for the overall system. Besides, the communication cost for tasks assigned to the same core is 0. (4) No preemption is allowed. That is, only one task is executed on one single core during one period of time, and one task can only be executed on one core at the same time. (5) We assume the decomposition and synthesis of tasks are performed by the cloud platform, and we only focus on the task assignment in this work.

Now we give the formulation of the task allocation problem. Suppose there are m resources $R = \{R_1, R_2, ..., R_m\}$ and n tasks $T = \{T_1, T_2, ..., T_n\}$. Let matrix $S_{n \times m}$ denote the execution time of each task on each resource:

$$S = \{ s_{i,j} \mid s_{i,j} \text{ denotes the execution time of } T_i \text{ on } R_j ; 1 \le i \le n, 1 \le j \le m \} . \quad (1)$$

Besides, let $E_{n \times m}$ be the resource allocation matrix, that is,

$$E = \{ e_{i,j} \mid e_{i,j} \text{ denotes if } T_i \text{ is allocated on } R_j ; \text{ if so, } e_{i,j} = 1, \text{ otherwise, } e_{i,j} = 0; 1 \le i \le n, 1 \le j \le m \} . \quad (2)$$

The objective is to find an optimal solution to achieve the best makespan and load balance. Suppose the sequence of task/resource is X, and the objective function can be defined as:

$$F(X) = \min\{ w_1 Makespan(X) + w_2 Load(X) \} , \quad (3)$$

where w_i denotes the weight of each dimension, and $\sum w_i = 1$, and the component in above equation adjusted so that the objective is minimum optimization. $Makespan(X)$ is the makespan of the whole task sequence:

$$Makespan(X) = \sum_{i=1}^{n} \sum_{j=1}^{m} C_1 s_{ij} e_{ij} , \quad (4)$$

and $Load(X)$ is the load factor of the resources:

$$Load(X) = \sqrt{\sum_{j=1}^{m} (1 - C_2 \frac{Y_j}{L_j})} , \quad (5)$$

where C_1, C_2 are constants, Y_j is the number of tasks assigned to resource R_j, and L_j is the computing capability of R_j.

Copyright ⓒ 2014 SERSC

4. Task Scheduling based on Modified ACO

4.1. Basics of ACO Algorithm

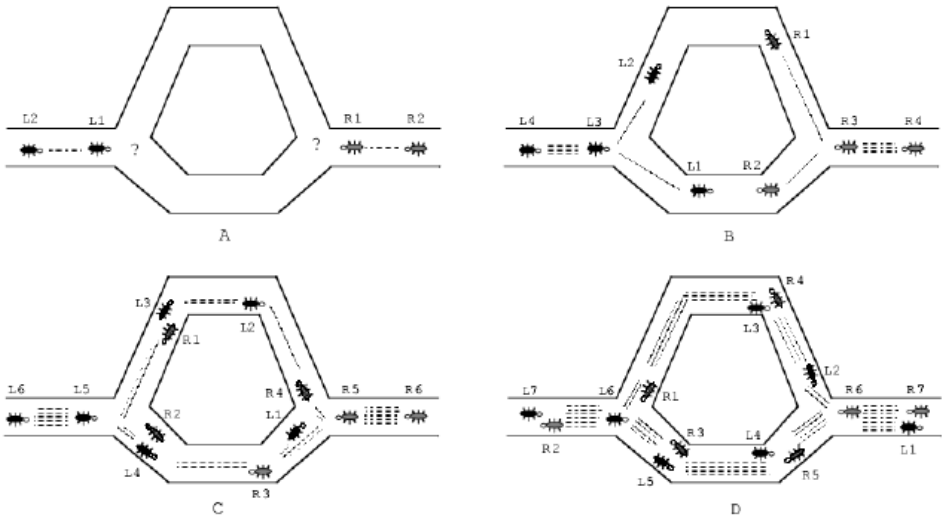

Figure 1. Illustration of ACO Algorithm

As shown in Figure 1, the principles of ACO can be explained through the double-bridge experiment conducted by Deneubourg et al. [23]. There are two designed routes in Figure 1-A, where the upper route is longer than the lower one. Two groups of ants are put at each end of the bridge, denoted as L and R respectively. At initial stage, since there is no pheromone heuristics, the decision at the cross is randomly made, as shown in Figure 1-B. After a while, more and more ants would select the lower route because it is shorter. Then, more pheromones are released on the lower route, which in turn increases the probability of being selected. Therefore, for new ants coming at the cross, they are more likely to choose the lower route, as shown in Figure 1-C and 1-D, where the dotted lines denote the amount of pheromones. Finally, with the accumulation of pheromones, almost all ants would be gathered on the lower route with the positive feedback mechanism.

The original ACO algorithm can be described as follows. Given a set of nodes $N = \{1,2,..., n\}$, the connections between nodes are denoted as $A = \{(i, j) \mid i, j \in N\}$, and the distance between nodes is expressed as a matrix $D = (d_{ij})_{n \times n}$. The initialized value of pheromone on each path (i, j) is $\tau_{ij} = \frac{1}{|A|}$, where A is the length of (i, j). Suppose there are m ants starting from node i_0.

The workflow of ACO is as follows. If the termination condition is satisfied, the algorithm ends and output the best solution. Otherwise, ant x starts from i_0, and $L(x)$ denotes the set of nodes x has passed. Initialize $L(x)$ as an empty set, and $1 \le x \le m$. When ant x arrives at node i, if $L(x) = \{l \mid (i,l) \in A, l \notin L(x)\}$ is empty, the journal of x ends. Otherwise, let $I = \{l \mid (i,l) \in A, l \notin L(x)\} - \{i_0\}$. If I is not empty, then x would transfer to node j with the probability of p_{ij}:

 Copyright ⓒ 2014 SERSC

$$p_{ij}(t) = \begin{cases} \dfrac{\tau_{ij}(t)^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \in I}\tau_{ij}(t)^{\alpha}\eta_{ij}^{\beta}}, & \text{if } j \in I; \\ 0, & \text{otherwise.} \end{cases}, \tag{6}$$

Where α is the heuristic factor indicating the importance of path with remaining pheromones, and β is the heuristic factor denoting the affect of heuristic information between steps. And $L(x)$ is updated as $L(x) \cup \{j\}$. If I is empty, which means ant x returns back to i_0, then update i as i_0 and starts over. The next step is to update the local best solutions and pheromones on the path. Denote the fitness evaluation function as f, and the shortest path of m ants as y. If $f(L(y)) < f(W)$, where W is the last best solution, update W as $L(y)$, and update the pheromones as follows:

$$\tau_{ij}(k) = \begin{cases} (1-\mu_{k-1})\tau(k-1) + \dfrac{\mu_{k-1}}{|W|}, & (i,j) \in W; \\ (1-\mu_{k-1})\tau(k-1), & \text{otherwise.} \end{cases}. \tag{7}$$

where μ_k is the decay coefficient, and $0 < \mu_k < 1$. In this way, the amount of pheromones on W are increased, and the pheromones on other routes are decreased, so that the best solution is discovered. Algorithm 1 shows the process of ACO.

4.2. Modified ACO for Task Scheduling on Heterogeneous Multi-core Processors

However, there are some limitations on traditional ACO algorithms, such as long searching time, local optimum and the stagnation phenomenon. Moreover, the capability in a heterogeneous multi-core processors environment is full of diversity. To this end, we propose to introduce the idea of immune algorithm into ACO to suit our task scheduling problem. Figure 3 shows the overall flow chart of our modified ACO algorithm.

4.2.1. Coding: The length of chromosome is set as the number of tasks n, and the genes on chromosome denote the sequence of task/resource pairs. For example, chromosome $\{1,1,2,5,..., m\}$ means that task T_1 is assigned to R_1, T_2 is assigned to R_1, T_3 is assigned to R_2, T_4 is assigned to R_5, and lastly T_n is assigned to R_m. The set of tasks assigned to R_j is denoted as $Y_j = \{T_1, T_2, ..., T_i\}$.

Copyright ⓒ 2014 SERSC

Algorithm 1 Ant Colony Optimization (ACO) Algorithm

Initialize the nodes $N = \{1, 2, \ldots, n\}$, resources $M = \{1, 2, \ldots, m\}$, ants $X = \{1, 2, \ldots, x\}$, connections $A = \{(i, j) | i, j \in N\}$, initial pheromone $\tau_{ij} = \frac{1}{|A|}$, and best solution $W = \{1, 2, \ldots, n\}$;
while Termination condition is not met **do**
  for Each ant x **do**
    Initialize $L(x) = \emptyset$;
    for Each node i in N **do**
      Set $L(x) = \{l | (i, l) \in A, l \notin L(x)\}$;
      if $L(x)$ is not empty **then**
        Calculate the transfer probability to next node j, p_{ij};
        Update $L(x) = L(x) \cup j$;
      end if
    end for
  end for
  Denote the route of shortest path at current iteration as y;
  Compute the fitness $f(L(y))$;
  if $f(L(y)) < f(W)$ **then**
    Update $W = L(y)$;
    Update pheromone on each path τ_{ij};
  end if
end while

Figure 2. Algorithm Description of Original ACO

4.2.2. Initialization: As mentioned before, the number of cores and computing capability of each processor are different. Therefore, we initialized the pheromone of each multi-core processor as follows:

$$\tau_j(0) = c_j \times MIPS_j, \tag{8}$$

where c_j is the number of cores on R_j and $MIPS_j$ is the computing capability of each core.

4.2.3. Vaccine extraction: In this step, ants with desirable qualities are acquired, and extracted as vaccine at some probability. The vaccine is used to improve the diversity of the population, in order to avoid local optimal solutions. The probability of ant x on resource i being selected as vaccine is:

$$p(x_i) = \frac{\sum_{j=1}^{m} |s_i - s_j|}{\sum_{j=1}^{m}\sum_{i=1}^{m} |s_i - s_j|}, i = 1, 2, \ldots, m, \tag{9}$$

where $s_j = \sum_{i=1}^{m} s_{ij}$ denotes the total time of all tasks executed on resource j.

 Copyright ⓒ 2014 SERSC

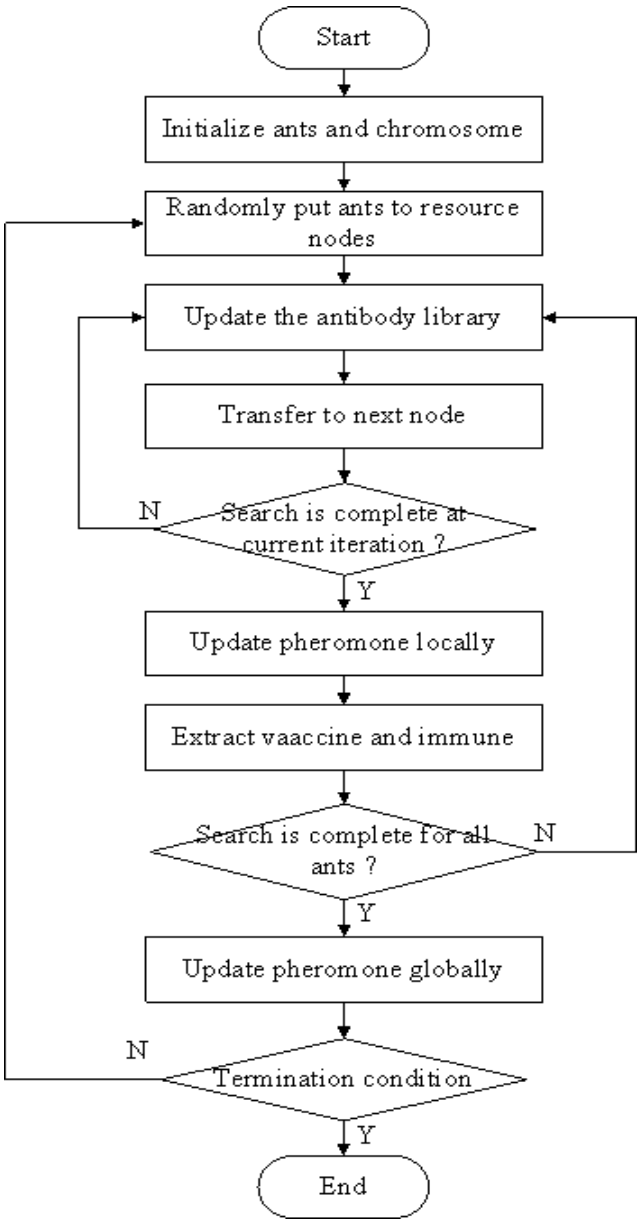

Figure 3. Flow Chart of Modified ACO

4.2.4. Transitivity: The transfer probability at iteration t is calculated as Equation (6). In order to avoid the local optimal solution, we introduce a threshold of pheromone. That is, if the amount of pheromone $\tau_{ij}(t)$ is no more than τ_0, the algorithm does not consider the influence from other nodes. Besides, we adjust μ if there is no improvement of the local solutions within N iterations:

$$\mu_{t+1} = \begin{cases} 0.95\ \mu_t, & \text{if } 0.95\ \mu_t \geq \mu_{\min}; \\ \mu_{\min}, & \text{otherwise.} \end{cases} \tag{10}$$

where $\mu_{\min}$ is the minimum value of μ.

4.2.5. Immune: Perform immune operation by integrating the extracted vaccine into the gene coding.

Copyright ⓒ 2014 SERSC

4.2.6. Update Pheromone: Let $\tau_j(t)$ be the amount of pheromone on resource j at time t, and we have

$$\tau_j(t+1) = (1-\mu)\tau_j(t) + \mu\Delta\tau_j, \tag{11}$$

where $\mu \in (0,1)$.

5. Experiment

5.1. Settings and Metrics

We use CloudSim [24] for simulating the heterogeneous multi-core processors environment, and each resource is actually a virtual machine. The proposed algorithm is implemented in MATLAB software. The parameter settings are as follows: $X = 50$, $\alpha = 1, \beta = 5, \mu = 0.6$.

In our experiments, we compare our modified ACO algorithm with original ACO and Round Robin (RR) algorithms. We design the comparison experiments with different CCR (Communicate-to-Computation-Ratio), calculated as:

$$CCR = \frac{\sum_{i=1}^{n}\sum_{i=1}^{m} c_{ij}}{\sum_{i=1}^{n} avg_i(s_{ij})}, \tag{12}$$

where c_{ij} is the communication cost between tasks, and $avg_i(s_{ij})$ is the average execution time of task i on different resources.

Beside, we employ the following metrics for evaluation.

(1) The scheduling length ratio (SLR), calculated as:

$$SLR = \frac{makespan}{CP}, \tag{13}$$

where $makespan$ is the overall execution time of all tasks, and CP is the critical path. The smaller SLR is, the better the scheduling algorithm is.

(2) Speedup, used to describe the parallel performance of task execution:

$$Speedup = \frac{\min\{\sum s_{ij}\}}{makespan}, \tag{14}$$

where s_{ij} is the time cost of task i on resource j, and $\min\{\sum s_{ij}\}$ is the time cost of sequential execution of all tasks. The larger $Speedup$ is, the more efficient the

 Copyright ⓒ 2014 SERSC

algorithm would be.

(3) Speed of convergence, which is a significant metric for swarm intelligence algorithms. The less iterations for global best solution, the more efficient the algorithm is.

5.2. Experimental Results

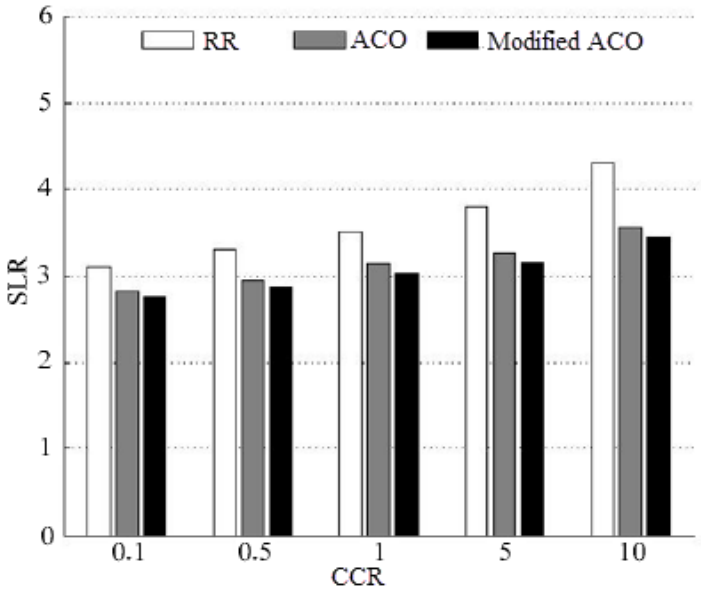

Figure 4. Results of SLR Comparison

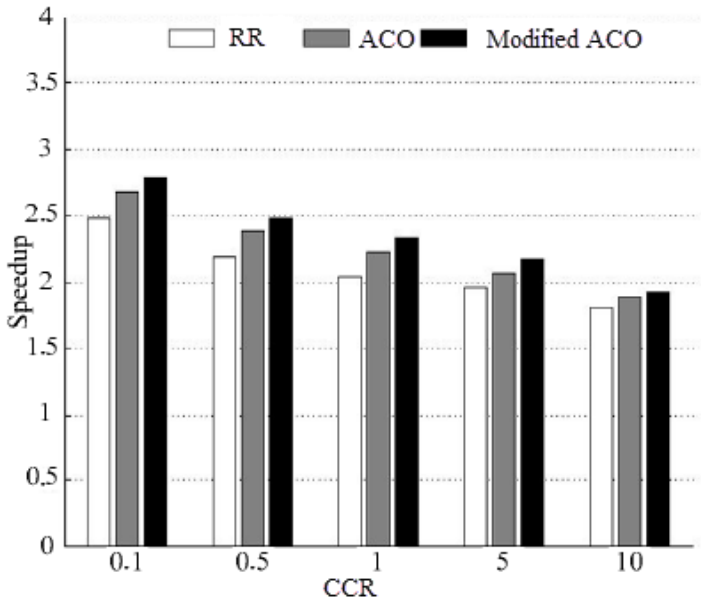

Figure 5. Results of Speedup Comparison

Figure 4 shows the results of SLR for three algorithms. We can see that as the CCR increases, the value of SLR grows as well. The reason is that the more communication cost there is between tasks, the more execution time it costs for the scheduling. Furthermore, our modified ACO algorithm outperforms others, and RR performs worst. We can also conclude that employing swarm intelligence optimization method is efficient

Copyright ⓒ 2014 SERSC

for task scheduling.

As shown in Figure 5, the speedup for all three algorithms decreases when the CCR goes big. Indeed, more communication affects the parallelism of the whole system. Similarly, we can also observe that our modified ACO has the best speedup performance.

Figure 6 gives the results of convergence of our modified ACO and ACO algorithms. We can see that our modified ACO converges after approximately 40 iterations, while the ACO converges after 50 iterations. Therefore, our modified algorithm has faster speed of convergence compared to the original ACO algorithm.

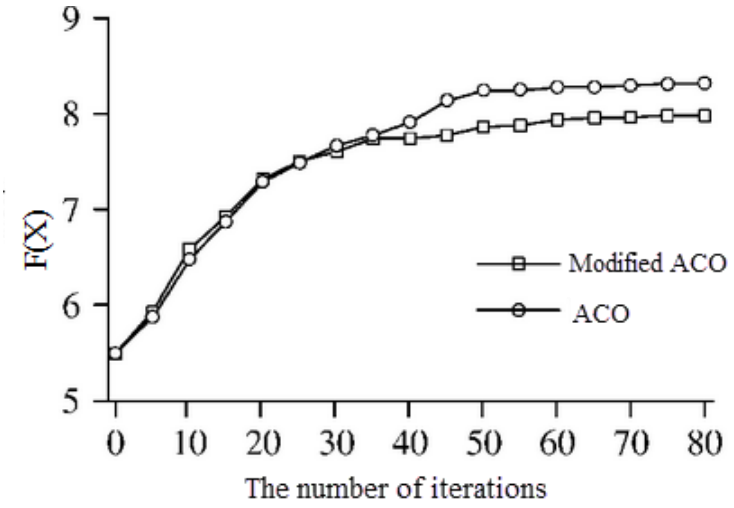

Figure 6. Results of Convergence

6. Conclusion

In this paper, we propose a modified Ant Colony Optimization (ACO) algorithm with immune algorithm, and apply it on the task scheduling problem in heterogeneous multi-core processors environment. However, we focus on the static scheduling in this work. That is, we assume the number of resource nodes is determined beforehand, and the computing capability of each processor is deterministic as well. In future, we will try to extend the algorithm for dynamic task scheduling, and apply it to elastic computing.

Acknowledgements

The research work was supported by the Fundamental Research Funds for the Central University under Grant no. N120323009, and the Doctoral Fund of Northeastern University at Qinhuangdao under Grant no. XNB201301.

References

[1] Z. Paul, and C. Eaton,“Understanding big data: Analytics for enterprise class hadoop and streaming data”, McGraw-Hill Osborne Media, **(2011)**.
[2] V. Kumar,*et al.*,“Introduction to parallel computing”,vol. 110,**(1994)**.
[3] P. Gepner and M. F. Kowalik, "Multi-core processors: New way to achieve high system performance", Parallel Computing in Electrical Engineering, PAR ELEC 2006. International Symposium on. IEEE, **(2006)**.
[4] K.Lakshmanan,, S. Kato and R. Rajkumar. "Scheduling parallel real-time tasks on multi-core processors", Real-Time Systems Symposium (RTSS), 2010 IEEE 31st. IEEE, **(2010)**.
[5] J. L. Gustafson, "Reevaluating Amdahl's law." Communications of the ACM, vol. 31, no. 5,**(1988)**, pp. 532-533.
[6] H. Luo ,D.J. Mu ,Z.-Q. Deng, X.-D. Wang, “A Review of Job Scheduling for Grid Computing”, Application Research of Computers, vol. 22, no.5,**(2005)**, pp.16-19.

 Copyright ⓒ 2014 SERSC

[7] M. Dorigo, M. Birattari, and T. Stutzle. "Ant colony optimization." Computational Intelligence Magazine, IEEE vol. 1, no. 4,**(2006)**, pp. 28-39.
[8] L.-N.Xing, *et al*., "A knowledge-based ant colony optimization for flexible job shop scheduling problems." Applied Soft Computing, vol. 10, no. 3,**(2010)**, pp. 888-896.
[9] K.-L. Huang and C.-J. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem", Computers & Operations Research, vol. 35, no. 4,**(2008)**, pp. 1030-1046.
[10] SiriluckLorpunmanee, *et al.*, "An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment", International Journal of Computer & Information Science & Engineering, vol. 1, no. 4,**(2007)**.
[11] V.Sarkar,"Partitioning and scheduling parallel programs for multiprocessors", MIT press, **(1989)**.
[12] T. L.Adam, K. M. Chandy and J. R. Dickson. "A comparison of list schedules for parallel processing systems." Communications of the ACM, vol. 17, no. 12,**(1974)**, pp. 685-690.
[13] I. Ahmad and Y.-K. Kwok, "A new approach to scheduling parallel programs using task duplication", Parallel Processing, ICPP 1994, International Conference,vol. 2, **(1994)**.
[14] T.Yang and A. Gerasoulis, "DSC: Scheduling parallel tasks on an unbounded number of processors", Parallel and Distributed Systems, IEEE Transactions, vol. 5, no. 9,**(1994)**, pp. 951-967.
[15] C. H.Papadimitriou and M. Yannakakis, "Towards an architecture-independent analysis of parallel algorithms", SIAM journal on computing, vol. 19, no. 2,**(1990)**, pp. 322-328.
[16] M. Seo and D. Kim, "Ant colony optimisation with parameterised search space for the job shop scheduling problem", International Journal of Production Research, vol. 48, no. 4,**(2010)**, pp. 1143-1154.
[17] P. Zhang, J. Lin, and L. Xue, "An Adaptive Heterogeneous Multiple Ant Colonies Algorithm", Journal of Intelligent Systems, vol. 19, no. 4,**(2010)**, pp. 301-314.
[18] I. Ellabib, P. Calamai, and O. Basir, "Exchange strategies for multiple ant colony system." Information Sciences, vol. 177, no. 5,**(2007)**, pp. 1248-1264.
[19] Y. Yang and M. S. Kamel, "An aggregated clustering approach using multi-ant colonies algorithms", Pattern Recognition, vol. 39, no. 7,**(2006)**, pp. 1278-1289.
[19] K. Ghoseiri and B. Nadjari, "An ant colony optimization algorithm for the bi-objective shortest path problem", Applied Soft Computing, vol. 10, no. 4,**(2010)**, pp. 1237-1246.
[20] C. Zhao, *et al.*, "Independent tasks scheduling based on genetic algorithm in cloud computing", Wireless Communications, Networking and Mobile Computing, WiCom'09. 5th International Conference on. IEEE,**(2009)**.
[21] L. Guo,*et al*., "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm", Journal of Networks, vol. 7, no. 3,**(2012)**.
[22] J.-F. Li, *et al*., "A task scheduling algorithm based on improved ant colony optimization in cloud computing environment", Energy Procedia, vol. 13,**(2011)**, pp. 6833-6840.
[23] J-L.Deneubourg, *et al*., "The self-organizing exploratory pattern of the argentine ant", Journal of insect behavior, vol. 3, no. 2,**(1990)**, pp. 159-168.
[24] R. N.Calheiros, *et al*., "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience, vol. 41, no.1,**(2011)**, pp. 23-50.

Authors

Zhulin Li, he was born in 1976. He received the M.S. degree from college of information science and engineering, Northeastern University, Shenyang, China. Now he is a Senior Experimentalist, and his research interests include task scheduling of CPU, secure programming.

Cuirong Wang, she was born in 1963. She received the Ph.D. degree from the school of information science and engineering, Northeastern University, Shenyang, China in July2004. Now, she is a professor and a research manager of the wireless sensor network and next generation network technology group. Her research interests include Routing Protocol, Network Security and Wireless Sensor Networks.

Copyright ⓒ 2014 SERSC

Haiyan Lv, she was born in 1979. She received Ph.D. degree from Chinese Academy of Agricultural Sciences. Now, she is a lecturer, and her main research fieldsare Remote Sensing and Geographical Information System.

Xin Song, she was born in 1978. She received Ph.D. degree from the school of information science and engineering, Northeastern University, Shenyang, China. Now, she is an Associate Professor, and her research interests include wireless sensor networks, distributed computing and cross layer optimization technology.

Copyright ⓒ 2014 SERSC