An Adaptive Immune System Applied to Task Scheduling on NOC

Wei Gao, Yubai Li, Song Chai, Jian Wang
School of Commutation and Information Engineering,
University of Electronic Science and Technology of China
Chengdu China, 611731
e-mail: wgeaio@163.com

***Abstract*—In this paper, an ADaptive Immune Algorithm (ADIA) based on the artificial immune system (AIS) is proposed for the dependent task scheduling on Network on Chip (NoC). We increase the diversity of population of AIS in two ways, and improve the output result of scheduling. On one hand, new calculation method of the number of clones and the probability of mutation are proposed to enlarge the search space appropriately. On the other hand, our algorithm adopts an adaptive strategy to enable the local search to drive the population evolution when evolution speed is low after many iterative processes. Moreover, the influences of different factors in the proposed algorithm are studied. The comparative simulation results show that our algorithm always outperforms the tradition algorithms.**

Keywords-component; AIS; Dependent Task Scheduling; Network on Chip; Population Diversification; Adaptive

I. INTRODUCTION

Network-on-Chip (NoC) is one of the most important technologies emerged in current multiprocessor systems. With the development and progress of the related technologies, NoC becomes the focus and advances technology in the related fields. The NoC is applied to solve many complex systems, such as in LTE [1] [2] [3]. As the solution of NoC has been more popular, researchers have focused on the task scheduling problem in NoC. To achieve the optimum solution, many heuristics are proposed, such as genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], ant colony optimization (ACO) [6], and simulated annealing (SA) [7].

In this paper, we propose an adaptive approach to scheduling real-time tasks based on an artificial immune system (AIS) for NoCs. Our work performs population-based optimization in order to find an optimal schedule. It extracts the knowledge from an individual which determines the evolution of the population including clone selection, clone mutation, individual extinction, and evolution speed. To drive the evolution speed, the adaptive strategy combined with the local search approach [8] is applied to our algorithm. Besides, given the trend that scheduling problem on NoCs not only focusing on the minimization of the completion time of all tasks, but also increasingly concerning with use efficiency of resources, multi-objective scheduling is also considered in our paper, including scheduling length (makespan), load balance [9] and average link load [10].

The rest of this paper is organized as follow: Section II reviews the related work in this area; Section III presents our proposal in detail; experimental results are shown in Section IV and Section V concludes the paper.

II. RELATED WORK

Recently, there have been many population-based algorithms of solving real-time task scheduling problem on NoC-based MPSoC, such as genetic algorithm (GA) and artificial immune system (AIS). In [4], GA is applied to task scheduling in multiprocessor system, in which GA shows its robust performance. Reference [11] proposed an energy efficient static algorithm based on genetic algorithm which optimizes the energy consumption of task communication in NoCs. In [12], it investigates genetic algorithm for static task scheduling in wormhole NoC-based systems to make all tasks and communication meet their deadlines.

For the artificial immune system, Reference [13] proposes an efficient method of extracting knowledge when scheduling parallel programs onto processors using an artificial immune system. In [14], a combinatorial optimization algorithms based on artificial immune systems is proposed to minimize the completion time. Reference [15] designs an algorithm based on artificial immune system to scheduling for heterogeneous computing environments.

III. THE PROPOSED ALGORITHM

This section presents a detailed description of the proposed ADIA, including the algorithm flow and various features borrowed from AIS and the local search.

A. Framework

The algorithm flow of the proposed algorithm is depicted in Figure 1. The population is randomly initialized, and each individual in the population, namely antibody, represent a solution for task scheduling. After the population initialization, the task priority of each task is calculated using its bottom level [4]. Following the task prioritization, each individual is evaluated and ranked by the fitness value calculated by equation (1), then a set of immune operations, including clone selection and mutation, is performed. To this point, the system estimates if the evolution speed is lower than a threshold. If the test result is yes, LS is then applied to each individual in the

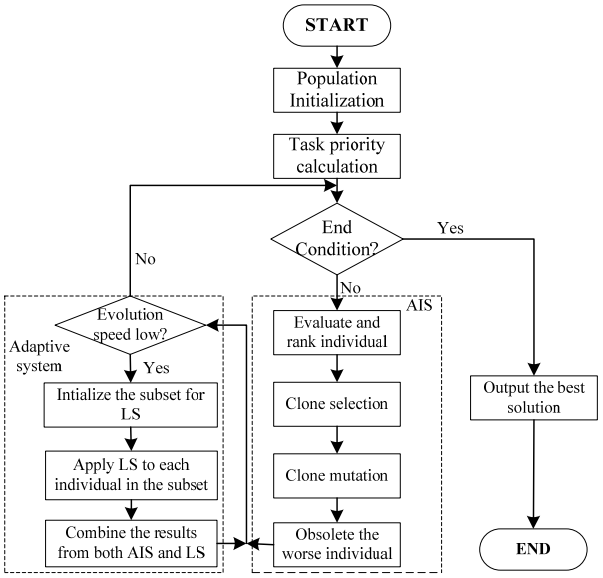

Figure 1. Flowchart of the ADIA

Selected subset to generate new better individuals. Combining the result of AIS with the result of LS, the new generation is generated.

B. Encoding of Solutions

The population in ADIA consists of a group of antibodies and each antibody represents a candidate assignment solution to the given scheduling. Antibodies are encoded as the strings of integers. Cell-indexes of the string depict task numbers and value of each cell of string represents the processor which task is assigned to. Suppose there are T tasks and P processors, so each cell of the string has the number between 0 and P-1. The example of an antibody for one problem with 10 tasks and 5 processors is illustrated in figure 2.

C. Diversification Using AIS

1) The affinity of antibodies

AIS acts upon a population of antibodies, which are modified by the affinity. The antibodies which have higher affinity are cloned, and the number of clones is proportional to the affinity of the antibody. Meanwhile, the probability of mutation for clones is inversely proportional to the affinity of the antibody. In other words, the affinity is critical to AIS. In our work, we regard the rank of the individual as the affinity for this individual. Firstly, the algorithm computes the fitness for each individual by simulating the execution of the program with the allocation encoded in the individual. Then, we rank individuals in the population according to the fitness. The

Antibody

3	1	2	0	3	2	4	0	2	4
T0	T1	T2	T3	T4	T5	T6	T7	T8	T9

Figure 2. The example of an antibody

fitness is the evaluation of the schedule solution, and is described as follow:

$$fitness = w_1 * \overline{makespan} + w_2 * \overline{load\_balance} + w_3 * \overline{ave\_link\_load} \quad (1)$$

where $\overline{makespan}$, $\overline{load\_balance}$, $\overline{ave\_link\_load}$ are normalized results, makespan is the time span for the NoC to complete all tasks.

The load balance measures the inverse coefficient of the total workload on each processor p_i. The larger load balance value suggests better balanced schedule. The load balance is defined as follow:

$$load\_balance = \frac{ave\_load}{\sum_{p_i \in P} (workload(p_i) - ave\_load)^2} \quad (2)$$

The average link load monitors the traffic load on each link, and is defined to be the average value of traffic loads on all links.

The largest fitness means the highest affinity for the individual, and the rank of the individual is assigned to be 0.

2) Clone selection

Selection is performed in every generation based on the affinity of antibodies. The expected number of times C_i for the antibody i being selected is inversely proportional to its rank r: $C_i=C_0 / (r+1)$ where C_0, the number of clones for the best individual, is a parameter of the operator. In our work, the calculation method of clone number differs from other artificial immune algorithms [13] in that we expanse the global search space with the same C_0, which is useful to diversify the population.

3) Mutation

All selected clones undergo the mutation process, the mutation ratio of the individual i is proportional to the rank r of the clone's parent. The probability of mutation is given by $p=round(P_0 \bullet (0.2+\Delta P \bullet r) \bullet n)$, where P_0 is the average probability of the mutation for the best parent, increased by ΔP for each following individual. Then the algorithm randomly selects p cells and sets them to a random value. In [13], $p=round(P_0 \bullet (1+\Delta P \bullet r) \bullet n)$, there the number of mutation of the better individual is apparently more. Then, we select clones with better affinity to keep evolving. Note that partial individuals that have worse fitness value are obsolete in the evolution processor.

D. The Adaptive System

Given the evolvement is based on the random method; the speed of evolution is also random. The evolution speed is the difference between the fitness of the best child and the fitness of the best parent. Our work drives them to evolve rapidly, when the evolution speed of the population is lower than a threshold. The main steps of the adaptive approach are described in Algorithm 1.

The number of individuals in the subset is determined by the parameter sampling_rate. The neighborhood solution x' is generated by randomly swapping positions

of two genes in the antibody. After completing the local search, the algorithm combines the results of local search with the results of AIS. Then the new generation has been generated, they will keep evolution till meeting the end condition.

Algorithm1:flow of the adaptive method

1. calculate the evolution speed of the AIS offspring
2. if the speed is lower than the threshold do
3. initialize the subset to undergo the local search processor
4. for each solution x in the subset do
5. repeat
6. k = 0
7. while k < k_{max} do
8. examine a neighborhood solution x' of x
9. calculate the fitness of x', if the evolution speed of x' is reach the requirement, then x = x'; otherwise set k ++
10. end while
11. end for
12. combine subset with the offspring from AIS
13. end if

IV. EXPERIMENTS AND RESULTS

A. Test Bed

In order to test the performance of the proposed ADIA algorithm, task graphs that are treated as the benchmark problems are generated by TGFF [16]. In this paper, there are four graph sizes, including 40, 60, 80, and 100. Meanwhile, each size contains 100 different graphs with similar characteristics which can be achieved with different seeds. Parameters values for TGFF are summarized in table I .

Each task graph is tested with same settings of *communication-to-computation* (CCR) [17] and processor number. To evaluate the performance of our proposal, we implement the scheduling algorithm in C++, and simulate the produced schedules under a SystemC based cycle-accurate NoC simulator.

TABLE I. PARAMETER SETTINGS FOR TGFF

Size	40	60	80	100
task_cnt	(32,1)	(50,1)	(70,1)	(90,1)
series_wid	(4,2)	(4,2)	(4,2)	(4,2)
series_len	(3,2)	(3,2)	(3,2)	(3,2)

B. Comparison Parameter Setting

The comparison result is presented with the ratio between two comparative algorithms, which is benefit for us to observe the performance difference. There are some common parameters in all comparative algorithms such as population size and weights in the fitness, where population size is 1000 and w_1=0.8, w_2=0.1, w_3=0.

In our proposal, there is a wide range of parameter settings, which are summarized as: C_0=100, P_0=0.1, ΔP=0.5, *sampling_rate*=0.3, and k_{max}=4. To keep pool size the same, the number of obsolete individuals is equal to the increase of evolving individuals. In GA, selection rate is 45%, crossover rate is 40% and mutation rate is 15%.

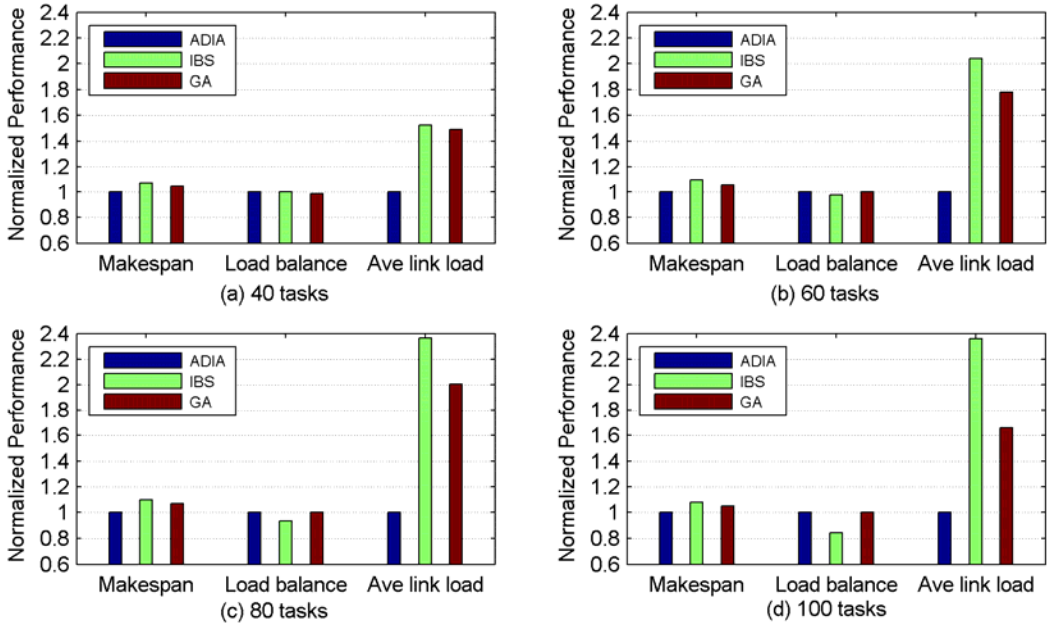

Figure 3. Comparative result with other algorithm

C. Experimental Results

The performance of the proposed algorithm is compared with those of the other two state-of-art scheduling algorithms including IBS in [13] and GA in [4]. In our paper, we only adopt the learning mode in [13] to compare with the proposed algorithm, because our study only focuses on learning mode.

The experimental results are described in figure 3. For each scale of task graph, every data in figure is the average of one performance objective ratio from 100 graphs. The optimal solution is obtained, when algorithms cannot find a better solution in consecutive 20 iterations in the population which has undergone evolution at least 40 generations. In the figure, all the results are normalized to the ADIA correspondence.

It is obvious that our proposed ADIA consistently outperforms IBS and GA in all test cases. On one hand, compared to other algorithms, the schedule provides by ADIA finishes the whole tasks faster. Taking the normalized makespan with 40 tasks as an example, compared to IBS the scheduling length of ADIA is reduced by7%, and it is 5% shorten than GA. On the other hands, it is notable that the ADIA is able to take advantage of resources more effectively. With the scale of scheduling problem increasing, the performance of the ADIA is also remarkable.

The advantage of our algorithm is that its search ability is strengthened by combining AIS with LS, hence, the population diversity of ADIA is more abundant than other algorithms. Moreover, the makespan optimized by ADIA is always shorter than both IBS and GA. However, the actually makespan acquired in the SystemC based NoC simulator sometimes is larger than that of IBS and GA, which is the reason that why the performance of ADIA in the figure is not so preeminent.

D. Factor Analysis

In order to find the significant of the calculation of the mutation, experiments are made with two different methods: (1) $p=round(P_0 \bullet (0.2+\Delta P \bullet r) \bullet n)$, (2) $p=round(P_0 \bullet (1+\Delta P \bullet r) \bullet n)$. The former is our proposal, the latter is proposed in [13] and the results are showed in table II.

TABLE II. COMPARATIVE RESULTS FOR MUTATION METHODS

Graph Size	Formulation	Makespan	Load balance	Link Load
40	(1)	1469.4	1.5534	8.20
	(2)	1564.4	1.5534	12.32
60	(1)	1712.5	2.3122	10.88
	(2)	1866.7	2.1684	23.44
80	(1)	1963.2	2.9787	13.92
	(2)	2206.3	2.7204	33.80
100	(1)	2409.6	4.1673	20.44
	(2)	2585	3.2517	47.48

In the table, each data is the average value of one objective for each graph size. The unit of the makespan is cycle, and the unit of the link load is flit. Obviously, the first method can deliver the better solution and the improvement is larger. The difference between the is that the probability of the better individual in the former is lower. In other words, the individual is better; the probability of mutation should be smaller. So, based on the simulation results, our selection of the method of the mutation is more suitable for the scheduling problem in the NoC multiprocessor context.

V. CONCLUSION AND FUTURE STUDY

In this paper, a novel hybrid meta-heuristic-based approach for solving task scheduling problem with multi-objectives in the multiprocessor system has been proposed by means of incorporating AIS with LS. Generally, our approach is improved upon a tradition AIS in two aspects. First, by means of proper calculation method of clone selection and mutation, the evolution in a right direction is guided with a great probability. Second, an adaptive system is introduced to improve the algorithm efficiency. Both manners are useful to diversify the population.

The performance of our proposed algorithm is compared with two related algorithms: IBS, GA. In experimental results, it shows that our proposed approach consistently outperforms the other two schedulers in terms of three schedule objectives. Besides, that proper mutation has a big influence on the performance of ADIA has been proved. The next work will further more study ADIA, and apply it to solve other multi-objective scheduling problems.

ACKNOWLEDGMENT

Thanks to the support of the National Natural Science Foundation of China [61201005], the Fundamental Resear ch Funds for the Central Universities [ZYGX2012J001], the National Major Projects [2011ZX03003-003-04].

REFERENCES

[1] R. Prolonge and F. Clermidy, "Network-on-chip traffic modeling for data flow applications," in Proceedings of the 2013 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, 2013, p. 2. *(references)*

[2] P. Ou, et al., "A 65nm 39GOPS/W 24-core processor with 11Tb/s/W packet-controlled circuit-switched double-layer network-on-chip and heterogeneous execution array," in Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International, 2013, pp. 56-57.

[3] H. Xie, et al., "Single-Chip Multiband EGPRS and SAW-Less LTE WCDMA CMOS Receiver With Diversity," Microwave Theory and Techniques, IEEE Transactions on, vol. 60, pp. 1390-1396, 2012.

[4] O. Sinnen, et al., "Toward a realistic task scheduling model," Parallel and Distributed Systems, IEEE Transactions on, vol. 17, pp. 263-275, 20.

[5] A. Omidi and A. Rahmani, "Multiprocessor independent tasks scheduling using a novel heuristic PSO algorithm," in Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on, 2009, pp. 369-373.

[6] A. Tumeo, et al., "Ant colony optimization for mapping and scheduling in heterogeneous multiprocessor systems," in

Embedded Computer Systems: Architectures, Modeling, and Simulation, 2008. SAMOS 2008. International Conference on, 2008, pp. 142-149.

[7] H. Jiang, et al., "A hybrid algorithm of Harmony Search and Simulated Annealing for multiprocessor task scheduling," in Systems and Informatics (ICSAI), 2012 International Conference on, 2012, pp. 718-720.

[8] T. Davidović, et al., "Permutation-based genetic, tabu, and variable neighborhood search heuristics for multiprocessor scheduling with communication delays," Asia-Pacific Journal of Operational Research, vol. 22, pp. 297-326, 2005.

[9] Y. Fang, et al., "A task scheduling algorithm based on load balancing in cloud computing," in Web Information Systems and Mining, ed: Springer, 2010, pp. 271-277.

[10] N. Nikitin, et al., "Physical-aware link allocation and route assignment for chip multiprocessing," in Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on, 2010, pp. 125-134.

[11] D. Shin and J. Kim, "Power-aware communication optimization for networks-on-chips with voltage scalable links," in Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, 2004, pp. 170-175.

[12] A. Racu and L. S. Indrusiak, "Using genetic algorithms to map hard real-time on NoC-based systems," in Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on, 2012, pp. 1-8.

[13] G. Wojtyla, et al., "Artificial immune systems applied to multiprocessor scheduling," in Parallel Processing and Applied Mathematics, ed: Springer, 2006, pp. 904-911.

[14] A. Costa, et al., "Makespan minimization on parallel processors: an immune-based approach," in Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on, 2002, pp. 920-925.

[15] H. Yu, "Optimizing task schedules using an artificial immune system approach," in Proceedings of the 10th annual conference on Genetic and evolutionary computation, 2008, pp. 151-158.

[16] K. Vallerio, "Task graphs for free (tgff v3. 0)," User's manual, 2003.

[17] A. K. Singh, et al., "Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms," Journal of Systems Architecture, vol. 56, pp. 242-255, 2010.