

An Improved Genetic Algorithm on Task Scheduling

Fangyuan Zheng^(✉) and Jingmei Li

College of Computer Science and Technology, Harbin Engineering University,
Harbin, China
zhengfangyuan@hrbeu.edu.cn

Abstract. Efficient task scheduling algorithm is critical for achieving high performance in heterogeneous multi-core processors. Because the existing genetic algorithm converges to local optimal solution, so an improved genetic algorithm is proposed to solve the above problems in this thesis. Firstly, the initial population is generated randomly according to the task height value, and then adopting the selection strategy based on competition scale. Finally, the crossover and mutation probability is improved to avoid premature phenomenon. The experiment based on randomly generated graphs shows that the proposed algorithm can improve the efficiency of convergence.

Keywords: Task scheduling · Heterogeneous multi-core processor
Genetic algorithm · Optimal solution

1 Introduction

With the development of computer architecture, chip multi-processor (CMP) [1] becomes the mainstream architecture and provides a platform for high-performance computing. In order to play the parallelism of CMP fully, a good task scheduling algorithm is very important.

Many scholars at home and abroad have carried out many studies on task scheduling, which has been proved to be NP complete [2]. Based on the above research, a new improved genetic algorithm (NIGA) for heterogeneous CMP is proposed, which improves the initial population mode, selection strategy, crossover and mutation probability. The experimental results show that the performance of NIGA is better than genetic algorithm (GA).

2 New Improved Genetic Algorithm

The GA can search the solution in parallel, but it also has the problems of premature and poor stability [3]. In response to the above shortcomings, NIGA is proposed to optimize GA.

2.1 Encoding and Decoding of Chromosomes

Chromosome encoding [4] mode is substring, a substring represents a processor core and each substring contains the number of task which is assigned to the same processor in sequence. Figure 1 is an example of chromosome encoding.

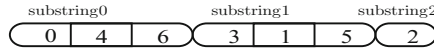


Fig. 1. An example of chromosome encoding

Chromosome decoding and encoding corresponds to each other, decoding is assigning tasks in sequence on the substring of chromosome to the corresponding processor core, then the structure of corresponding task scheduling is constructed.

2.2 Population Initialization and Fitness Function

The individual generation strategy in population is to randomly assign tasks to different processor cores. The tasks on the same core are sorted by the task height. The sequence of tasks with same height is generated randomly. The task height is defined as Eq. (1).

$$h(N_i) = \begin{cases} 0, & \text{if } pre(N_i) = \phi \\ \max_{N_j \in pre(N_i)} \{h(N_j)\} + 1 & \text{else} \end{cases} \quad (1)$$

In NIGA, the quality of individuals is measured with fitness value, the individual with larger value has greater probability to be selected into next generation, and with smaller will be eliminated after some operations. The calculation of fitness is shown in Eq. (2).

$$f(X_i) = \frac{1}{SL(X_i)} \quad (2)$$

In Eq. (2), $SL(X_i)$ represents the scheduling length of individual X_i .

2.3 Selection

After the population initialization, the selection strategy is used to select several individuals randomly, then choosing individual with the highest fitness value to the next step. The difference between initial individuals is large, only the smaller competition scale can guarantee the population diversity. With the individual quality becomes better, the scale becomes larger in order to search the optimal solution globally. The strategy sets the scale double by 20 times, as shown in Eq. (3).

$$K = 2 \times \frac{t}{20}, \quad t \in T \quad (3)$$

2.4 Crossover and Mutation

The main function of crossover is to generate new individual, the mutation operation is mainly to maintain species diversity [5]. If crossover and mutation probability is too large, some individuals with better fitness may be destroyed, it is not conducive to the solution convergence; if the probability is too small, it may not produce new individuals. Therefore, the probability should be adaptive, which can be changed with the fitness value, so as to ensure that individuals with low fitness value have a large probability, and the individuals with high fitness value has a small probability to save excellent individuals. The probability is shown in Eq. (4).

$$P(i) = \begin{cases} P_{\max} & f_i \leq f_{avg} \\ P_{\min} + (P_{\max} - P_{\min}) \cot[\frac{\pi}{4} (\frac{f_i - f_{avg}}{f_{\max} - f_{avg}} + 1)] & f_i > f_{avg} \end{cases} \quad (4)$$

In Eq. (4), P_{\min} and P_{\max} is the minimum and maximum of crossover and mutation probability, f_{\max} , f_{avg} , f_i is the maximum, average and i -th individual of fitness value respectively.

2.5 Termination Conditions

Set the maximum evolution number T_{\max} , NIGA is stopped after the certain iterations, then the individual with maximum fitness value is the optimal task scheduling.

3 Experiments

Randomly generated DAG is used as input data, by comparing the scheduling length and algorithm convergence to measure NIGA and GA.

The communication calculation rate (CCR) of DAG is 0.5 and the processor number is 3. The initial calculation parameters of GA and NIGA are: population size $M = 100$, maximum iterations $T_{\max} = 200$. Moreover, P_c of GA is 0.7, P_m is 0.02, the crossover P_{\min} and P_{\max} of NIGA are 0.8 and 0.2 respectively, the mutation P_{\min} and P_{\max} are 0.03 and 0.002 respectively. In order to avoid the randomness, the average of 15 experimental results is used as the test result of scheduling length.

The experiment mainly tests the scheduling length of GA and NIGA with different nodes, the result as shown in Table 1.

Table 1. The scheduling length of GA and NIGA with different nodes

Algorithm	Nodes = 10	Nodes = 20	Nodes = 30
GA	76	104	137
ICLGA	65	89	116

The experiment mainly tests the iterative evolution on same task number (Task Nodes = 20) of GA and NIGA algorithm, the experimental results are shown in Fig. 2.

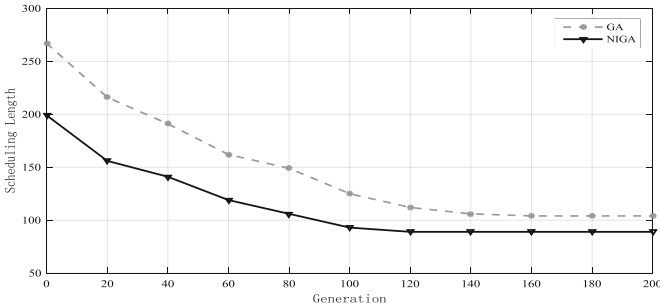


Fig. 2. The iterative evolution of GA and NIGA

From Table 1 and Fig. 2, it can be concluded that the scheduling length of NIGA is shorter than that of GA with the same task nodes, that is, the optimal solution of NIGA is the best, the time of optimal solution is shorter and the convergence speed is faster.

4 Conclusion

The NIGA algorithm is a better task scheduling algorithm based on heterogeneous CMP. It overcomes the shortcomings of GA and improves the scheduling efficiency. NIGA improves the initial population, uses the fitness selection strategy, adopts adaptive crossover and mutation probability to promote the global optimal solution. The experimental results show that the NIGA algorithm has the highest quality of the optimal solution and is faster than GA algorithm.

References

1. Keshanchi, B., Souri, A., Navimipour, N.J.: An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. *J. Syst. Softw.* **124**, 1–21 (2016)
2. Akkasi, A.: Genetic algorithm for task scheduling in heterogeneous distributed computing system. **6**(7) (2015)
3. Ahmad, S.G., Liew, C.S., Munir, E.U., et al.: A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems. *J. Parallel Distrib. Comput.* **87**(C), 80–90 (2015)
4. Ahmad, S.G., Munir, E.U., Nisar, W.: PEGA: a performance effective genetic algorithm for task scheduling in heterogeneous systems. In: *International Conference on High Performance Computing and Communication, 2012 IEEE International Conference on Embedded Software and Systems*. IEEE, pp. 1082–1087 (2012)
5. Singh, R.: An optimized task duplication based scheduling in parallel system. **8**(8), 26–37 (2016)