Genetic Algorithm Approach to Schedule Tasks

Ashish Mittal[1], Kapil Chawla[2]
[1, 2] SITM, Sonipat

Abstract- *Task scheduling is the allocation of resources over time to perform a collection of tasks. Task Scheduling in Multiprocessor is a term that can be stated as finding a schedule for a general task graph to be executed on a multiprocessor system so that the schedule length can be minimized. Multiprocessor scheduling problems may be divided in two categories: Static and dynamic task scheduling. A static or deterministic task scheduling is one in which precedence constraints and the relationships among the task are known well in advance While non-deterministic or dynamic scheduling is one in which these information is not known in advance or not known till run time. Efficient scheduling of application tasks is critical to achieving high performance in parallel multiprocessor systems. The choice of scheduling technique and its parameterization impacts the performance of systems. This paper describes an algorithm for scheduling tasks to multiple processors. The algorithm is "Genetic Algorithm". Genetic algorithm (GA) is a meta-heuristic technique and also a search technique used to find approximate solutions to optimization and search problems*

Keywords- Task Scheduling, Optimization, Multiprocessor, Genetic Algorithm etc.

I. INTRODUCTION

Task Scheduling in Multiprocessor is a term that can be stated as finding a schedule for a general task graph to be executed on a multiprocessor system so that the schedule length can be minimized. Multiprocessor scheduling problems can be classified into many different categories based on characteristics of the program and tasks to be scheduled, the multiprocessor system, and the availability of information. A major factor in the efficient utilization of multiprocessor system is the proper assignment and scheduling of computational tasks among processors.

The problem can have many variations:

(a) The scheduling algorithm can be deterministic – also known as static – or nondeterministic: A deterministic task scheduling problem is defined as one in which the knowledge related to tasks, their relations towards each other, timing and the number of processors used are all a-priori knowledge. In a nondeterministic problem on the other hand, all or some of these factors can be input-dependent and vary according to run time conditions.

(b) The tasks can be preemptive or non-preemptive: A preemptive task scheduling problem allows the tasks to be cut off from execution and another task to begin or continue its execution cycle [operating system example]. A non-preemptive problem in which task execution must be completely done before another task takes control of the processor.

(c) The processors can be either homogenous or heterogeneous: Heterogeneity of processors means that the processors have different speeds or processing capabilities. In a homogenous environment on the other hand, all processors are assumed to have equal capabilities.

II. PROPOSED ALGORITHM

A) GENETIC ALGORITHM STRUCTURE

Typically, a genetic algorithm consists of the following steps:

- GA1: Initialization – initialize the population.
- GA2: Evaluation – evaluate each chromosome using fitness function.
- GA3: Genetic operations –Select the parent and apply genetic operators on them to produce new chromosomes.
- GA4: Repeat steps GA2 and GA3 until termination condition reached.

From the above steps, we can see that genetic algorithms utilize the concept of survival of the fittest; passing "good" chromosomes to the next generation.

B) STRUCTURE OF THE CHROMOSOMES

We define our chromosome structure as a combination of two strings SQ and SP, whose length equal to the number of tasks. SQ (scheduling queue) maintains precedence constraints between tasks, and an entry in SQ represents a task to be scheduled. An entry in SP (scheduling processor) represents the processor to the corresponding task is scheduled onto.

The details to generate a chromosome can be seen in following steps:

- IP1: Select randomly a task from the entire entry tasks. Set this task as the first task in SQ.
- IP2: Repeat step IP3 for (v-1) times.
- IP3: Randomly select a task who is not in SQ and whose predecessors all have been in SQ, and add this task to SQ.
- IP4: For SP part, randomly generate an integer number between 1 and m for each task in SQ and add it to SP.

C) EVALUATION

In order to select good chromosomes, we define the fitness function as:

F(i) = (maxFT-FT(i) +1)/ (maxFT-minFT+1)

Where: maxFT and minFT is the maximum and minimum finishing time of chromosomes in current generation, respectively. FT (i) is the finishing time of the ith chromosome. Once the fitness values of all the chromosomes have been evaluated, we can select the higher fitness value chromosomes.

Average Fitness: After calculating the fitness values of all chromosomes, average values will be calculated by summing up the fitness values of all chromosomes divided by population size.

avgF= sum of fitness of all chromosomes/pop_size

Where avgF is the average fitness of the current generation and pop_size is the population size.

III. REPRODUCTION: CROSSOVER AND MUTATION

A) CROSSOVER

As our chromosome comprises two separate parts SP and SQ having different characteristics, we will employ crossover policies for SQ part. Here, initially the population size is 20. After sorting, out of 20 chromosomes, mutation is applied on the first 10 best chromosomes (based on their fitness values i.e. higher fitness values chromosomes) and crossover operator is applied on the remaining 10 chromosomes.

Details about crossover are given in following steps:

CR1: Set the Crossover probability.
CR2: Select the two adjacent chromosomes after doing mutation on first 10 best chromosomes. The crossover is applied on the adjacent chromosomes for example on the 11^{th} and 12^{th} chromosomes and then 13^{th} and 14^{th} chromosomes up to 19^{th} and 20^{th} chromosome.
CR4: Generate two crossover points P and Q between 1 and v for both chromosomes. For example, p=1 and q=5.
CR5: Rearrange the order of tasks in SQ between p and q of one chromosome according to the order of tasks of another chromosome, and put the resulting chromosomes after crossover in next generation.

B) MUTATION

Mutation can be considered as a random alternation of the individual. Details about mutation are given in following steps:

MT1: Set the Mutation probability.
MT2: After sorting all the chromosomes based on their higher fitness values, mutation is applied on the first 10 best chromosomes. Firstly, select the first chromosome after sorting and this is the best chromosome having higher fitness value.
MT3: Generate two mutation points, p (tasks Ti) and q (Task Tj) between 1 and v for the selected chromosome. For example p=3 and q=8.
MT4: Form a new chromosome by exchanging the two point p and q in the chromosome.

IV. CONCLUSION

The problem of scheduling of tasks to be executed on a multiprocessor system is one of the most challenging problems computing. Genetic algorithms are well adapted to multiprocessor scheduling problems. As the available resources are increased to the GA, it is able to find better solutions. GA performs better as compared to other traditional methods. Overall, the GA appears to be the most flexible algorithm for problems using multiple processors. It also indicates that the GA is able to adapt automatically to changes in the problem to be solved.

REFERENCES

[1] G. Syswerda and J. Palmucci, “The application of genetic algorithms to resource scheduling”, Proceedings of the Fourth International Conference on Genetic Algorithms and Their Applications, pages 502-508, San Mateo, CA, July 1991.

[2] Goldberg, David E, “Genetic Algorithms in Search, Optimization and Machine Learning”, Kluwer Academic Publishers, Boston, 1989.

[3] Mitchell, Melanie, "An Introduction to Genetic Algorithms", MIT Press, Cambridge, MA. 1996.

[4] L.M.Schmitt, "Fundamental Study Theory of Genetic Algorithms", International Journal of Modelling and Simulation Theoretical Computer Science. 2001.

[5] C. V. Ramamoorthy, "Optimal scheduling strategies in a multiprocessor system", IEEE Trans. Computers, vol. C-2I.,Feb. 1972.

[6] E. S. H. Hou, R. Hong, and N. Ansari, "Efficient multiprocessor scheduling based on genetic algorithms", IEEE 1990.

[7] Muhhamad K. Dhodhi, Imtiaz Ahmad, Ishfaq ahmad, "A multiprocessor scheduling scheme using Problem-space genetic algorithms", IEEE 1995.

[8] Ali Pedram, "A method for scheduling multi processing systems with genetic algorithm", International Journal of Engineering and Technology Vol. 1, No. 2, June, 2009.

[9] Intisar A.Majied Al-Said, Nedhal Al-Saiyd, Firas Turki Attia, "Multiprocessor scheduling based on genetic algorithms", 2009.

[10] Sachi Gupta, Gaurav Agarwal, "Task Scheduling in Multiprocessor System Using Genetic Algorithm", Second International Conference on Machine Learning and Computing, IEEE 2010.

[11] Amir Masoud Rahmani and Mojtaba Rezvani, "A Novel Genetic Algorithm for Static Task Scheduling in Distributed Systems", 2009.

[12] Carnegie-Mellon, "Genetic Algorithms and Their Applications", Proc. of the First Int. Conference, July 24-26, 1985.

[13] Dr. Franz Rathlauf, "Representations for Genetic and Evolutionary Algorithms", 2nd edition, @ Springer. 2006.