Role of Genetic Algorithms in Task Scheduling Optimization

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

Abstract- *Task scheduling in multiprocessor systems also known as multiprocessor scheduling has been a source of challenging problems for researchers in the area of computer engineering. The general problem of multiprocessor scheduling can be stated as scheduling a set of partially ordered computational tasks onto a multiprocessor system so that a set of performance criteria will be optimized. Efficient scheduling of application tasks is critical to achieving high performance in parallel multiprocessor systems. Genetic algorithm is a meta-heuristic technique and also a search technique used to find approximate solutions to optimization and search problems. The major steps involved are the generation of a population of solutions, finding the objective function and fitness function and the application of genetic operators. In this Paper, following things are being discussed, task scheduling, Role of Genetic Algorithm in task scheduling in multiprocessor environment, why we need this algorithm, when we can use this algorithm, Genetic operators.*

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

I. INTRODUCTION

Genetic algorithms operate on finite-sized populations of candidate schedules. At each iteration of the algorithm, relatively poor schedules are removed from the population and are replaced with new candidate schedules generated by: (1) applying mutations to individual schedules in the population; (2) applying cross-over operations to pairs of schedules in the population.

Genetic algorithms as powerful and broadly applicable stochastic search and optimization techniques, are the most widely known types of evolutionary computation methods today. The father of the original Genetic Algorithm was John Holland who invented it in the early 1970's. In general, a genetic algorithm has five basic components as follows:

(i) An encoding method that is a genetic representation (genotype) of solutions to the program.
(ii) A way to create an initial population of individuals [Davis, 1991].
(iii) An evaluation function, rating solutions in terms of their fitness, and a selection mechanism.
(iv) The genetic operators (crossover and mutation) that alter the genetic composition of offspring during reproduction.
(v) Values for the parameters of genetic algorithm

II. LITERATURE REVIEW

(I) EFFICIENT MULTIPROCESSOR SCHEDULING BASED ON GENETIC ALGORITHMS [E. S. H. HOU, R. HONG, AND N. ANSARI] [1990]. In this paper [10], the authors proposed an efficient method based on genetic algorithms to solve the multiprocessor scheduling problem. The representation of the search node will be based on the schedule of the tasks in each individual processor. The genetic operator proposed is based on the precedence relations between the tasks in the task graph. The proposed genetic algorithm will be applied to the problem of scheduling the robot inverse dynamics computations. The genetic operator developed takes into account the precedence relations of the tasks and guarantee that the new strings generated are legal.

(II) A MULTIPROCESSOR SCHEDULING SCHEME USING PROBLEM-SPACE GENETIC ALGORITHMS [MUHAMMAD K. DHODHI, IMTIAZ AHMAD, ISHFAQ AHMAD] [1995]. In this paper [11], the authors proposed a technique based on Problem-space genetic algorithms (PSGA). PSGA combines the search power of GA with List scheduling heuristics in order to reduce the completion time and to increases the resource utilization. In the following paper, a PSGA based technique is proposed for static multiprocessor scheduling problem including the communication delays to reduce the completion time and to increase the throughput of the system.

(III) IMPROVED MULTIPROCESSOR TASK SCHEDULING USING GENETIC ALGORITHMS [MICHAEL BOHLER, FRANK MOORE, YI PAN] [1999]. This paper [12] described the design and implementation of a genetic algorithm for minimizing the schedule length for a general task graph to be executed on a multiprocessor system. Minor changes to the program would easily support the introduction of interprocessor

communication delays and overhead costs of the system (Jesik et al 1997), as well as other options.

III. WHY GENETIC ALGORITHMS?

It is no surprise that GA is starting to find more common use in the workplace. With their inherent simplicity and versatility, they are often not difficult to implement onto existing applications. Genetic Algorithms are also very time efficient, capable of finding solutions to problems in a short amount of time.

(i) GA can solve hard problems quickly and reliably: Since GA are population-based algorithms that rely on sampling the given population, they are able to sustain reliable results with a predictable margin of error.

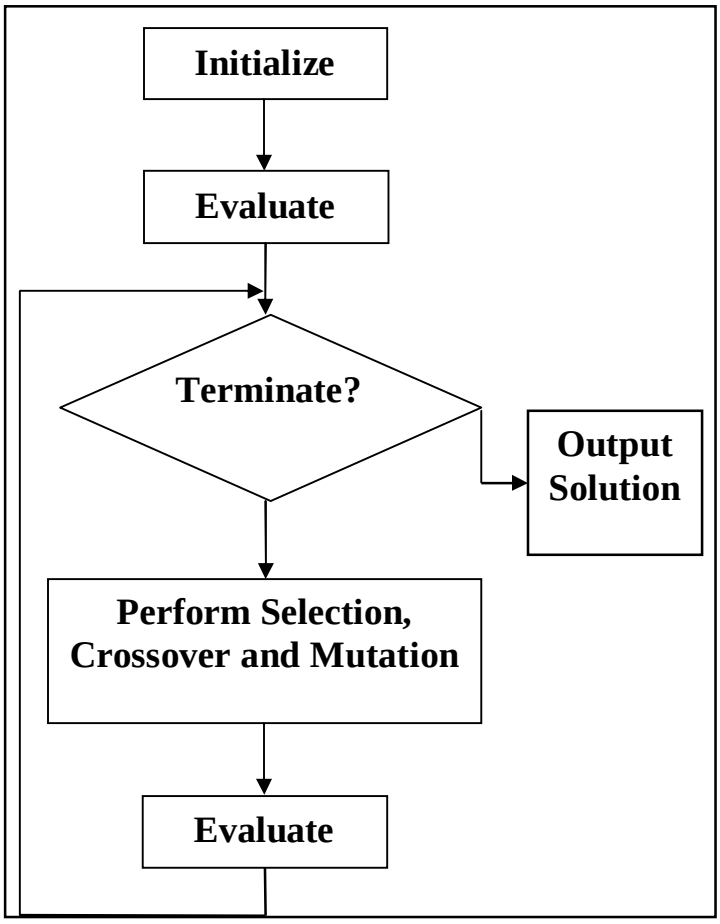
(ii) GA is easy to interface to existing simulations and models: Since GA require very little problem specific information, they are relatively easy to adapt to existing applications and models. The only thing that a GA needs is to have a candidate solution passed into it and to return the result of its fitness test.

(iii) GA is extensible: In a given population, the solution spaces may be highly multi modal; there may be many possible global solutions in the population. As in reality, there may not be just one global solution for a given problem. An example is the population patterns of certain species. There may be more than one area in the world where this specie may thrive. There are various niches instead of one localized area. GA account for this possibility through modifying the selection process. This allows for the testing of more than one strong solution area.

(iv) GA is easy to hybridize: It is also possible to make a GA more effective by incorporating problem specific search techniques into the GA. Some applications may have better means of finding the best solution space than using selection and crossover. Incorporating these means in place of or in conjunction with selection and crossover may help yield positive results in a shorter span of time.

IV. FLOW CHART OF GENETIC ALGORITHM

This flowchart illustrates the basic steps in a GA. The steps shown in the flowchart are discussed briefly as:



(a) **Initialization:** The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated.

(b) **Evaluation:** Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions are evaluated.

(c) **Termination criteria:** The genetic algorithm uses three termination criteria.

1. The aim of the genetic algorithm is to find a solution for a problem. So when such a solution is found the genetic algorithm stops. One termination criterion is the fitness of the best genotype of the population. Hence the genetic algorithm terminates when the best genotype has reached or surpassed the target solution fitness.
2. Another method is to keep track of the number of fitness evaluation calls made by the genetic algorithm, function calls to determine the fitness of child genotypes. The genetic algorithm terminates after a set number of these evaluations are done.
3. A genetic algorithm can also be terminated at the end of every cycle, when manually interrupted by the user.

(d) **Selection:** Selection allocates more copies of those solutions with higher fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones, and many selection procedures have been proposed to accomplish this idea.

(e) **Recombination:** Recombination combines parts of two or more parental solutions to create new, possibly better solutions (i.e. offspring). There are many ways of accomplishing this, and competent performance depends on a properly designed recombination mechanism. The offspring under recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner.

(f) **Mutation:** While recombination operates on two or more parental chromosomes, mutation locally but randomly modifies a solution. Again, there are many variations of mutation, but it usually involves one or more changes being made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution.

(g) **Replacement:** The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generation-wise replacement and steady-state replacement methods are used in GAs.

(h) Repeat steps 2–6 until a terminating condition is met.

V. 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 resources are increased available 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.

The advantages of the GA approach are that it is simple to use, requires minimal problem specific information, and is able to effectively adapt in dynamically changing environments.

REFERENCES

[1] Yi-Wen Zhongiz, Jian-Gang Yang, "A Genetic algorithm for tasks scheduling in parallel multiprocessor systems", Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an, 2-5 November 2003

[2] Michael Rinehart, Vida Kianzad, and Shuvra S. Bhattacharyya, "A Modular Genetic Algorithm for Scheduling Task Graphs", 2003.

[3] M. Salmani Jelodar, S. N. Fakhraie, F. Montazeri, S. M. Fakhraie, M. Nili Ahmadabadi, "A Representation for Genetic-Algorithm-Based Multiprocessor Task Scheduling", Congress on Evolutionary Computation, Vancouver, BC, Canada, IEEE July 16-21, 2006.

[4] M. Nikravan and M. H. Kashani, "A Genetic algorithm for process scheduling in distributed operating system considering load balancing ", Proceedings 21st European Conference on Modelling and Simulation, 2007.

[5] Keshav Dahal, Alamgir Hossain, Benzy Varghese, "Scheduling in Multiprocessor System Using Genetic Algorithms", 7th Computer Information Systems and Industrial Management Applications, IEEE 2008.

[6] Davis, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, 1991.

[7] E. Hou, R. Hong, and N. Ansari, "Multiprocessor scheduling based on genetic algorithms", Dept of ECE, New Jersey Institute of Technology, Technical Report, Aug. 1990.

[8] Forrest, Stephanie. "Genetic algorithms: principles of natural selection applied to computation", Science, vol.261, 1993.

[9] Tang, K.S., K.F. Man, S. Kwong and Q. He. "Genetic algorithms and their applications", IEEE Signal Processing Magazine, vol.13, 2004.

[10] Michael Bohler, Frank Moore, Yi Pan. "Improved Multiprocessor Task Scheduling Using Genetic Algorithms", Twelfth International FLAIRS Conference, 1999.

[11] S.N.Sivanandam, S.N.Deepa, “Introduction to Genetic Algorithms”, Springer-Verlag Berlin Heidelberg, 2008.

[12] Forrest, Stephanie. "Genetic algorithms: principles of natural selection applied to computation" Science, vol.261. 1993.