International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 10, October- 2014

Task Scheduling in Homogeneous Multiprocessor Multi-Network Systems using Evolutionary Techniques

Aparna Vishwanath
Dept. of Electronics Engg.
FRCRCE,
Mumbai, India

Ramesh Vulavala
Dept. of Chemical Engg.
DJSCE,
Mumbai, India

Sapna U. Prabhu
Dept. of Electronics Engg.
FRCRCE,
Mumbai, India

Abstract: **Minimizing the total execution time of all tasks in a given network by scheduling on a multi-processor system is an important and challenging problem. The classical techniques of optimization require considerable time to address this problem. This problem gathers larger proportions as the number of task networks increases. This work proposes a genetic algorithm based task scheduling method for a multi-network multiprocessor system, with and without interleaving. The proposed method achieves reduced total processing time in both the cases. The proposed algorithm is tested by varying number of populations and crossover probability.**

Keywords- Directed Acyclic Graph, Multi- network, Multiprocessor, Total Processing Time, Edge-zeroing, Interleaving, Genetic Algorithms.

I. INTRODUCTION

The multiprocessor scheduling problem is generally stated as follows: Given a multiprocessor computing system and a specific number of tasks to execute, "how does one efficiently schedule the tasks to make optimal use of the computing resources "? [1].In general, a deterministic search of the solution space to identify an optimal solution to this NP-complete problem is computationally and temporally exhaustive [2]. The extent of this issue depends mainly upon the following factors: the number of tasks, execution time of the tasks, precedence order of the tasks, number of processors, their uniformity (homogeneous/heterogeneous) and inter-task communication.

In multiprocessor systems, factors like load balancing, and allocation of tasks onto different processors when they are heterogeneous, may also influence the overall performance. In this work, all the processors are assumed to be homogeneous and the load balancing takes into consideration the utilization factor of each processor.

As task scheduling in multiprocessor systems is a NP-complete problem, the classical techniques take a large amount of time to arrive at the optimal solution. Hence, in our work, we propose genetic algorithm as a technique to solve the scheduling problem in lesser time compared to the classical techniques.

In this work, the multiprocessor scheduling problem is considered as a parallel program represented by an directed acyclic task graph (DAG). Following are the assumptions made for the system under consideration:

- Tasks have precedence constraints.
- The period of the tasks, execution time of the tasks and the communication delay between the tasks executing on different processors are available as inputs to the system.
- The tasks in the system are assumed to be periodic and non-preemptive.
- The processors are assumed to be identical (homogeneous).

The rest of the paper is organized as follows: Section 2 gives a brief review of related work. Section 3 describes the preliminaries relevant to the work done. Proposed algorithm for the task scheduling problem is explained in Section 4. Section 5 provides simulation results, performance analysis followed by the conclusions.

II. RELATED WORK

S. H. Houet. al [3] implemented a genetic algorithm based task scheduling method for homogeneous multiprocessor systems using string representations. The strings are ascending order of tasks arranged with respect to the height values. An axiom that 'A schedule satisfying the height-ordering condition is a legal schedule' has been followed in their work. The total processing time of the system has been optimised. The results obtained are compared with list algorithm and optimal schedule generated for random task graphs. Kwok et. al. [4] suggested various static scheduling algorithms for allocating directed task graphs to multiprocessors. A detailed procedure of every algorithm with an example has been presented in their work. In our work we have used heuristics for initial population generation, as suggested in this paper. Kwok et. al. [5] proposed an efficient technique for scheduling task graphs to multiprocessors using parallel genetic algorithm. In their work the initial population is generated using various heuristics. Since the precedence order was to be preserved, a variant of crossover viz. order crossover has been used. Concept of adaptive probabilities for crossover and mutation operation have been used. The results have been recorded for single network scenario with variation in

communication to computation ratio. Yogesh R. Sahare [6] proposed a hybrid genetic algorithm for task scheduling in multiprocessor systems. The initial population has been generated using the earliest start value of each node in the task graph, a neighbourhood search technique has been used for selecting parent chromosomes with fitness value more than 75% of mean population fitness. The reason for doing so is stated as *'possibility of finding the best solution by performing genetic operations on these parents is higher than that of the rest of the population'.* Results have been recorded for optimised schedule with an effort to reduce the make-span of the system. The results obtained have been compared with Genetic Algorithm, Tabu search technique and simulated annealing. Ranjit Rajak et. al. [7] proposed a task scheduling method for homogeneous multiprocessor system using Fork-Joint method. The fork-joint mechanism has been used at every level of DAG. Firstly a fork or joint structure is identified, then the task with maximum fork or joint value is scheduled on to the same processor as that of the parent task provided all the precedence constraints are satisfied. The results obtained have been compared with the heuristic based algorithms. It has been concluded that their proposed method fetches lower values of total processing time compared to the heuristic based algorithms.

III. PRELIMINARIES

Genetic algorithms (GA) are search algorithms based on mechanics of natural selection and natural genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomised, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of GA are designed to simulate processes in natural systems necessary for evolution especially those follow the principles first laid down by Charles Darwin of "survival of the fittest."

The GA maintains a population of n chromosomes (solutions) with associated fitness values. Parents are selected to mate, on the basis of their fitness, producing offspring via a reproductive plan. Consequently highly it solutions are given more opportunities to reproduce, so that offspring inherit characteristics from each parent. As parents mate and produce offspring, room must be made for the new arrivals since the population size is kept at a static size. Individuals in the population die and are replace by the new solutions, eventually creating a new generation once all mating opportunities in the old population have been exhausted. In this way it is hoped that over successive generations better solutions will thrive while the least fit solutions die out.

After an initial population is randomly generated, the algorithm evolves through three operators [5]:

1. Selection (Reproduction)
2. Crossover
3. Mutation (optional)

Selection: Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected.

Crossover: Crossover is the GA's primary local search routine. The crossover/reproduction operator computes two offspring for each parent pair given from the selection operator. These offspring, after mutation, make up the new generation. A probability of crossover is predetermined before the algorithm is started which governs whether each parent pair is crossed-over or reproduced. Reproduction results in the offspring pair being exactly equal to the parent pair. The crossover operation converts the parent pair to binary notation and swaps bits after a randomly selected crossover point to form the offspring pair.

Mutation (optional): Mutation of a chromosome is achieved by simply flipping a randomly selected bit of the chromosome.

Compared to other existing evolutionary techniques viz. Ant Colony Optimization, Differential Evolution etc., Genetic Algorithms prove to be robust.

In the single network scenario [1], various available heuristics viz. t-level, b-level, sl-level, alap and random generation were used as suggested by Kwok and Ahmad [4] to generate the initial population for Genetic Algorithms.

t-level: The t-level of a node 'n' is the length of the longest path from an entry node to 'n' (excluding 'n').

b-level: The b-level of a node 'n' is the length of the longest path from node 'n' to an exit node.

sl-level: sl levels are static b-levels computed as (t-level – b-level).

alap: The ALAP (as late as possible) start-time of a node is a measure of how far the node's start time can be delayed without increasing the schedule length.

Genetic algorithms are best suited to continuous optimization problems. As the scheduling problem is discrete in nature with a severe constraint on maintaining the order sequence of tasks, it requires certain modifications as suggested by Kwok and Ahmad [5] and cannot be used in its original form.

IV. PROPOSED ALGORITHM

In the system under consideration, since the tasks have precedence constraints between them, the order of tasks needs to be maintained as given by the DAG. Thus, the aim of the work is to minimize the total execution time without violating the precedence order.

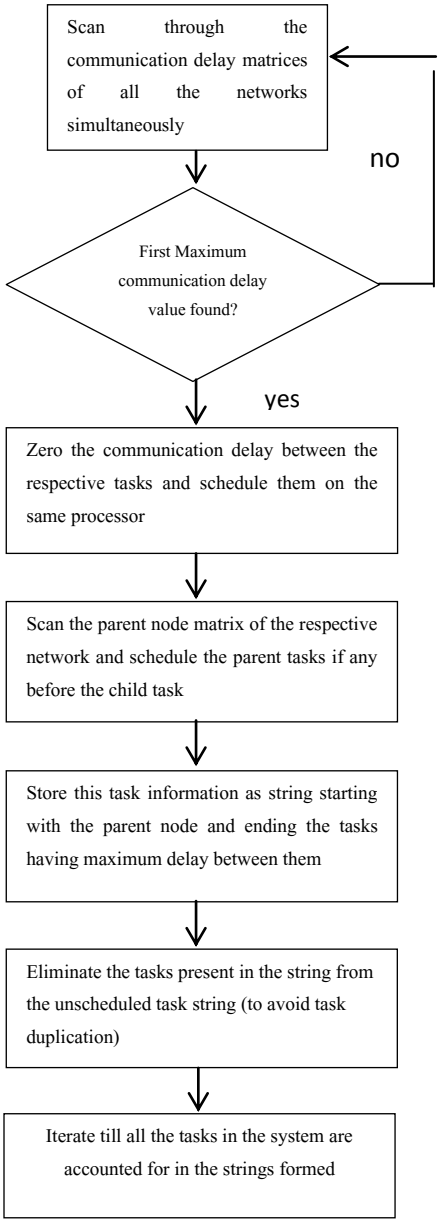
The proposed strategy is to tackle all the networks simultaneously.

The detailed description of the work done is as below:

Initial population generation:

The initial population for GA is formed by ordering the tasks along with their parent tasks in the descending order of their communication delays. The edge zeroing concept has been used in this algorithm which is a linear clustering technique.

The edge zeroing procedure is illustrated in the flowchart below:



The procedure starts by zeroing the edges having maximum communication delay between tasks. Edge zeroing is done by scanning through all the networks in the system simultaneously, finding the tasks having maximum communication delayand then zeroing it. However these can only be executed after completing their preceding tasks. The record of parent tasks of respective tasks in the task graphs are maintained as parent node matrices. Hence the predecessors are stored along with the maximum delay task in a string. The string is then removed from the multi-network system, and stored in a list. We then search for the next maximum delay task in the remaining set of tasks. This process is iterated till all the tasks become part of some string. These strings are treated as separate entities in forming the initial populations of GA. Since these strings satisfy the precedence order automatically, the normal operations of GA can be applied to them, without further modifications. Zeroing the edges dictates that the tasks are scheduled on the same processor.

The strings are then scheduled onto processors on the basis of Utilization Factor of the processor. The Utilization Factor (U_i) of a processor is given as,

$$U_i = e_i/p_i$$

Where,

e_i – execution time of the task

p_i – period of the task

When the processor utilization exceeds 80% (i.e. 0.8) scheduling starts from the next processor and so on. When tasks from the same string are scheduled onto different processors, the communication delay and idle time are added to the total processing time (TPT).

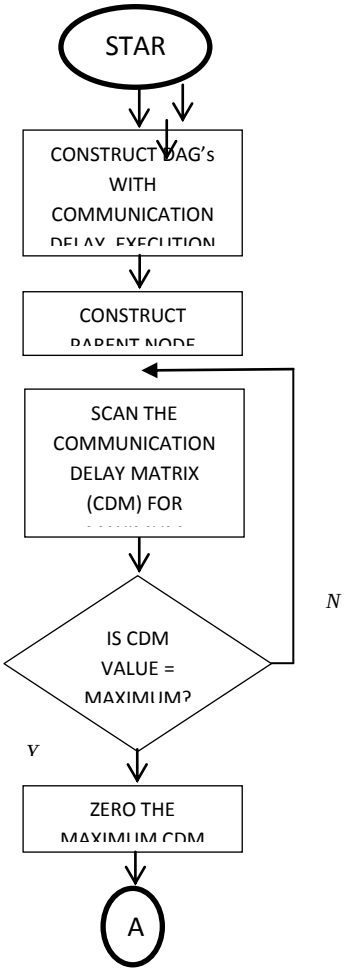
TPT (on different processors) $= \sum idle\ time + communication\ delay + execution\ time$

TPT (on same processor) = $\sum Execution\ times$

The fitness function (FF) of each member of the population is then calculated as the reciprocal of the total processing time(TPT).

$$\boldsymbol{FF = 1/(1 + TPT)}$$

In the selection process we ensure that the best sequence is passed on to the next generation. Other sequences are selected based on their fitness values. This is followed by crossover operation.



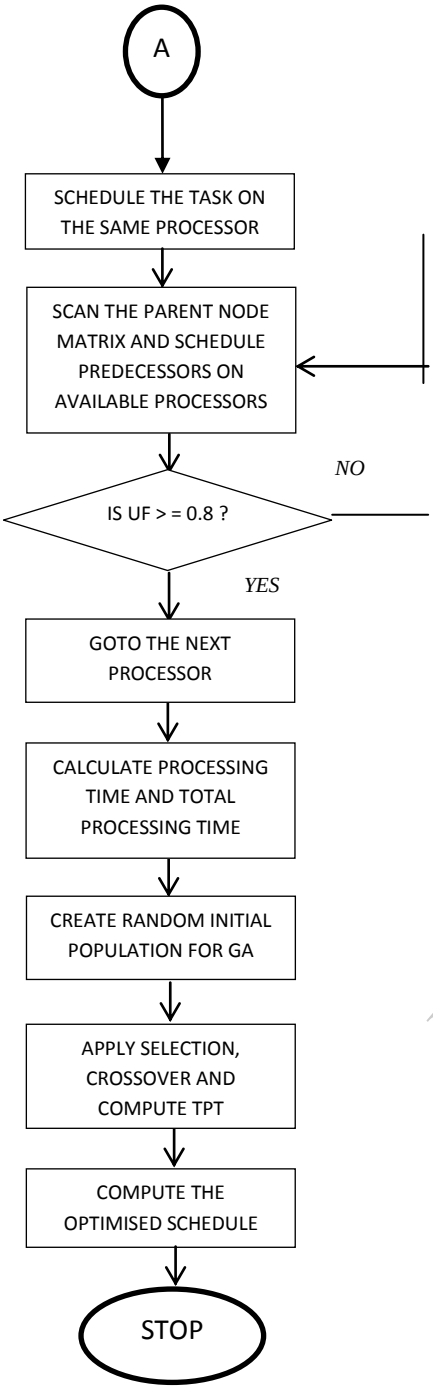


Figure 1 : FLOWCHART

As mentioned in Section 3, crossover occurs with a certain crossover probability called the crossover rate. In this work, adaptive probability is used as suggested by Srinivas et. al [8]. Figure 1 gives a better insight of the work done.

The adaptive crossover rate μ_c is defined as follows:

$$\mu_c = kc(fmax - f')/(fmax - favg)$$

The program developed is scalable and adaptable to the change in number of tasks and task graphs of a parallel processing system. The number of processors on the target multiprocessor system depends upon the utilization factor of processors, which is evaluated when scheduling takes place. Therefore, it can be said that the number of processor in the system depends upon the period and execution time of tasks. Other parameters that can be easily modified include the number of iterations of the genetic algorithm (number of generations) and population size.

V. RESULTS AND CONCLUSIONS

VARIATION IN NUMBER OF POPULATIONS (Np) :

Following are the results obtained for a scenario when networks are considered independently one at a time with variation in the size of populations.

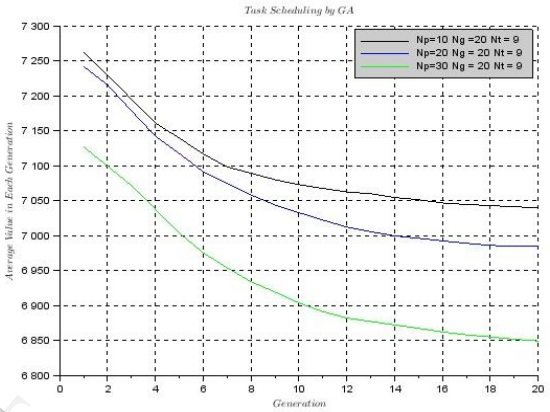


Figure 2: Single Network Scenario

Following are the results obtained for a scenario when networks are considered simultaneously (interleaving) with variation in size of populations.

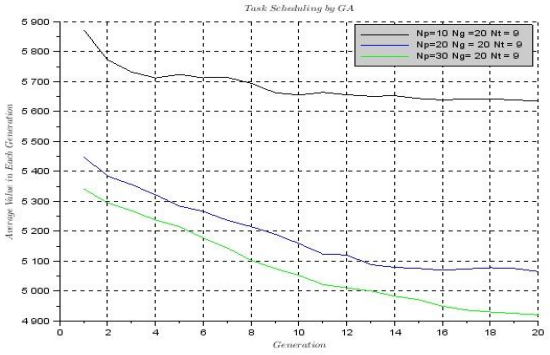


Figure 3: Interleaved Networks Scenario

In both cases, an increase in the population size decreases the average of total processing time, as expected. However, for a specific population size the proposed interleaving technique shows considerable improvement in minimizing the total processing time. This is because of the additional freedom available with the strings coming from all the networks of the system.

VARIATION IN CROSSOVER PROBABILITY (μc) :

Following are the results obtained for a scenario when networks are considered independently one at a time with variation in crossover probability value.

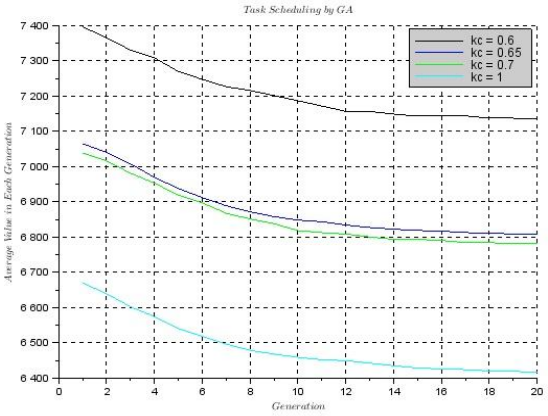


Figure 4: Average Total Processing Time with variation in kc for single network scenario

Following are the results obtained for a scenario when networks are considered simultaneously (interleaving) with variation in the value of crossover probability.

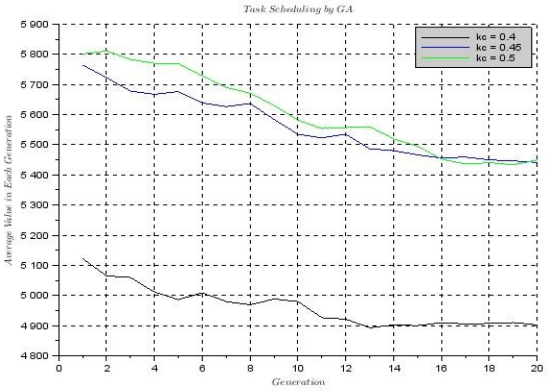


Figure 5: Average Total Processing Time with variation in kc for interleaved networks

From fig. 5 and 6 it is clearly seen that for higher values of kc in the single network scenario, we achieve better values of average processing time whereas for interleaved network case the lower the kc value the better the result. The reason possibly is that, the heuristic based initial population in case of interleaving seems to be very close to the optimal value requiring only minor adjustments from cross-over. Whereas in the case of the individual networks, the initial population seems to be far from optimal value depending heavily on cross-over to reach the final value.

FUTURE WORK

The same strategy can be easily extended to cyclic networks. It can also be used to investigate the performance of multiprocessor systems for mixed networks where some are periodic and some are not.

REFERENCES

[1] Aparna Vishwanath, Ramesh Vulavala and Sapna U. Prabhu,"Task Scheduling in Homogeneous Multiprocessor systems using Evolutionary Techniques", IJETAE, vol.4, issue 2, February 2014.
[2] Poonam Panwar, A.K. Lal, Jugminder Singh, "A Genetic Algorithm based Technique for Efficient Scheduling of Tasks on Multiprocessor System," Proceedings of International Conf. on SocPros 2011, AISC 131, pp.855-861.
[3] Edwin S. H. Hou, Nirwan Ansari and Hong Ren, ' A Genetic Algorithm for Multiprocessor Scheduling', IEEE Transactions on Parallel and Distributed Systems, Vol. 5, No. 2, Feb 1994.
[4] Yu-Kwong Kwok and Ishfaq Ahmad (1999), " Static Scheduling Algorithms for allocating directed task graphs to multiprocessors."
[5] Yu-Kwong Kwok and Ishfaq Ahmad (1997), "Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm
[6] Yogesh R. Shahare, ' Multiprocessor Task Scheduling Using Hybrid Genetic Algorithm' , ITSI Transactions on Electrical and Electronics Engineering (ITSI-TEEE), Vol. 1, Issue 4, 2013.
[7] RanjitRajak and C.P. Katti, ' Task Scheduling in Multiprocessor System using Fork-Joint Method (TSFJ), International Journal of New Computer Architectures and their Applications (IJNCAA), 2013.
[8] M. Srinivas and L.M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms," *IEEE Trans. Sys., Man and Cybernetics*, vol. 24, no. 4, Apr. 1994, pp. 656-667.
[9] Michael Bohler, Frank Moore, Yi Pan (1999), " Improved Multiprocessor Task Scheduling Using Genetic Algorithms" Proc. of Twelfth International FLAIRS Conference.
[10] E.G. Coffman, *Computer and Job-Shop Scheduling Theory*, Wiley, New York, 1976.
[11] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.
[12] J. Ullman, "NP-Complete Scheduling Problems," *J. Comp. Sys. Sci.*, 10, 1975, pp.384-393.
[13] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich., 1975.
[14] W. Atmar, "Notes on Simulation of Evolution," *IEEE Trans. Neural Networks*, vol. 5, no. 1, Jan. 1994, pp. 130-147.
[15] Probir Roy, Md. Mejbah U Alam and Nishita Das, (2012)," Heuristic based task scheduling in multiprocessor systems with genetic algorithm by choosing the eligible processor", International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.4.
[16] L.D. Davis (Ed.), *The Handbook of Genetic Algorithms*, New York, Van N. Reinhold, 1991.
[17] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass., 1989.
[18] R. Tanese, "Parallel Genetic Algorithm for a Hypercube," Proc. *Int'l Conf. on Genetic Algorithms*, 1987, pp. 177-183.
[19] "Distributed Genetic Algorithms," Proc. *Int'l Conf. Genetic Alg.*, 1989, pp. 434-439.
[20] M. Srinivas and L.M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms," *IEEE Trans. Sys., Man and Cybernetics*, vol. 24, no. 4, Apr. 1994, pp. 656-667.
[21] A. Srinivasan and J. Anderson, "Fair scheduling of dynamic task systems on multiprocessor", The Journal of Systems, vol. 77, pp. 67-80, 2005.