ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

A Module Scheduling for Multiprocessor System Based on Memetic Algorithms

Surinder Kumar
Department of Mathematics, S.G.G.S College, Sec-26, Chandigarh, India

Abstract: - In multiprocessor systems, an efficient scheduling of a parallel program onto the processors that minimizes the entire execution time is vital for achieving a high performance. The problem of multiprocessor scheduling can be stated as finding a schedule for a general module graph to be executed on a multiprocessor system so that the schedule length can be minimize. This scheduling problem is known to be NP- Hard. In multiprocessor scheduling problem, a given program is to be scheduled in a given multiprocessor system such that the program's execution time is minimized. The objective is making span minimization, i.e. we want the last job to complete as early as possible. The modules scheduling problem is a key factor for a parallel multiprocessor system to gain better performance. A module can be partitioned into a group of sub modules and represented as a DAG (Directed Acyclic Graph), so the problem can be stated as finding a schedule for a DAG to be executed in a parallel multiprocessor system so that the schedule can be minimized. This helps to reduce processing time and increase processor utilization. Genetic algorithm (GA) is one of the widely used techniques for this optimization. But there are some shortcomings which can be reduced by using GA with another optimization technique, such as simulated annealing (SA). This combination of GA and SA is called memetic algorithms. This paper proposes a new algorithm by using this memetic algorithm technique.

Keywords: - **Module Scheduling, Genetic Algorithm (GA), Simulated Annealing (SA), Memetic Algorithms, Directed Acyclic Graph (DAG).**

I. INTRODUCTION

Multiprocessors systems have emerged as a powerful computing means for real-time applications such as those found in nuclear plants and process control because of their capability for high performance and reliability. The problem of scheduling of real-time modules in multiprocessor systems is to determine when and on which processor a given module executes [21].

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. The main goal behind research on genetic algorithms is robustness i.e. balance between efficiency and efficacy

Simulated Annealing is a related global optimization technique which traverses the search space by testing random mutations on an individual solution. A mutation that increases fitness is always accepted. A mutation which lowers fitness is accepted probabilistically. SA can also be used within a standard GA algorithm, simply by starting with a relatively high rate of mutation, which decreases over time along a given schedule. Memetic Algorithm is a form of genetic algorithm that are combined with other forms of local search, such as simulated annealing.

Scheduling is to simply allocate a set of modules to resources such that the optimum performance is obtained. The purpose of scheduling is to distribute the modules among the processors in such a way that the precedence constraints are preserved and the total execution time T is minimized.

The value of T depends on the given allocation of modules and some scheduling policy applied to modules. The scheduling policy defines an order of processing modules, ready to run in a given processor.

II. PROBLEM STATEMENT

The goal of multiprocessor scheduling is to find an optimization algorithm to minimize the overall execution time for a collection of sub modules that compete for computation.

A. Given:

- A multiprocessor system with 'm' machines.
- A module represented by a DAG.
- The estimated execution duration of every sub module and the estimated data transmission duration between adjacent sub modules.

***Find*:**

A schedule to,

$$\text{Minimize Max [finish time (Vj)]}_{j=1}^{n}$$

where, the schedule determines, for each sub module, both the processor on which execution will take place and the time interval within which it will be executed [5] [9].

B. System Model

A (homogeneous multiprocessor system is composed of a set $P = \{p_1, p_2, ..., p_m\}$ of 'm' identical processors. They are connected by a complete communication network where all links are identical. Module pre-emption is not allowed. While computing, processor can communicate through one or several of its links [3] [5] [6] [10].

C. Problem Constraints

- Precedence relations among the modules exist. These determine the order of module execution.
- Communication cost (cost to send messages from a module on one processor to a succeeding module on a different processor) exist. The communication cost between two modules on the same processor is assumed to be zero.
- Module duplication is allowed i.e. same module may be assigned to more than one processor to

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

reduce communication costs and schedule length.

- The multiprocessor system consists of a limited number of fully connected processors [4].

D. Task Graph

The problem of optimal scheduling a module graph onto a multiprocessor system with 'p' processors is to assign the computational modules to processors so that precedence relations are maintained and all of the modules are completed in shortest possible time.

A schedule (fig.2) is a vector, $S = \{s_1, s_2, \ldots, s_n\}$, where $S_j = \{t_{i1}, t_{i2}, \ldots, t_{in_j}\}$ i.e. S_j is the set of the n_j modules scheduled to processor, p_j. The total execution time yielded by a schedule is called make span. The time when the last module is completed is called the finishing time (FT) of schedule [5].

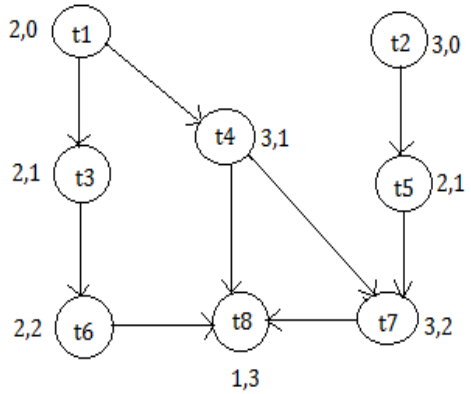

Fig 1: A Module Graph

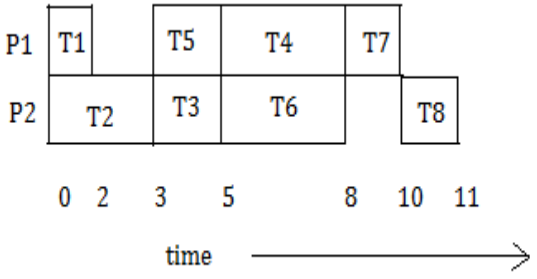

Fig 2: A schedule for two processors displayed as Gantt chart

The preceding schedule, corresponding to module graph (fig.1) has a finishing time of 11 units of time. An important lower bound for finishing time of any schedule is the critical path length.

Notations

For a module graph (fig1) TG = (V, E):

- T_i is a predecessor of T_j and T_j is a successor of T_i if $e_{ij} \in E$.
- T_i is an ancestor of T_j and T_j is a child of T_i if there is a sequence of directed edges leading from T_i to T_j.
- PRED(T_i) – The set of predecessor of T_i
- SUCC(T_i) – The set of successor of T_i
- $E_t(T_i)$ – The execution time of T_i.

Height of a task graph

The height of a module in a module graph is defined as-

$$\text{Height } (T_i) = \begin{cases} 0 & ,if\ PRED(T_i) = \phi \\ 1 + \max_{T_j \in PRED(T_i)} height(T_j) & ,otherwise \end{cases}$$

This height function indirectly conveys precedence relations between the modules. If the module Ti is an ancestor of module T_j, then height (T_i) < height (T_j).

If there is no path between the two modules, then there is no precedence relation between them and order of their execution can be arbitrary [2] [6].

E. String representation

For multiprocessor scheduling problem, a legal search node (a schedule) is one that satisfies following conditions.

1. The precedence relations among the modules are satisfied
2. Every module is present and appears only once in the schedule.

A schedule can be represented as several lists of computational modules (fig 3). Each list corresponds to computational modules executed on a processor and order of modules in the list indicates the order of execution.

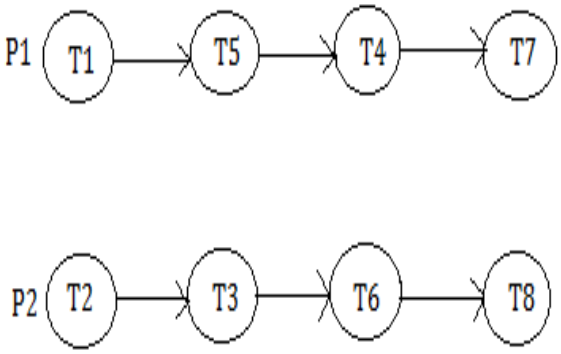

Fig 3: List Representation of A schedule.

This ordering allows us to maintain the precedence relations for the modules executed in a processor and ignore precedence relations between modules executed in different processors. Each list can be further viewed as a specific permutation of modules in the list.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

$$\begin{bmatrix}1&2&3&4&5&6&7&8\\5&3&6&7&4&8&1&2\end{bmatrix} = \begin{bmatrix}1&5&4&7\\5&4&7&1\end{bmatrix}\begin{bmatrix}1&5&4&7\\5&4&7&1\end{bmatrix}$$

Fig 4: Permutation Representation of Schedule.

This representation is useful when we actually implement the genetic algorithm [1] [2] [6].

III. METHODOLOGY

A. *Initial Population*

For genetic algorithm, a randomly generated initial population of search nodes is required. We impose the following height ordering condition on schedules generated:

"The list of modules within each processor of schedule is ordered in an ascending order of their height".

For example, consider module graph in the figure. Module T_5 (height=1) is an ancestor of T_8 (height=3). If they are assigned to same processor, then T_5 will precede T_8 according to the height ordering and this would guarantee that T_5 will be executed before T_8 in that processor. Similarly, T_6 and T_5 are not related and hence they can be executed in any order.

Algorithm to generate initial population

Algorithm Generate-Schedule // {Generates a schedule of module graph TG for multiprocessor System with 'p' processors}
1. [Initialize] Compute height for every module in TG
2. [Separate modules according to their height]
3. [Loop p-1 times] for each of first p-1 processors, do step 4
4. [Form the schedule for a processor]
5. [Last processor] Assign remaining modules in the set to last processor [2] [6] [16].

B. *Fitness Function*

It is the objective function that we want to optimize in the problem. It is used to evaluate the search nodes and also controls the genetic operators. For multiprocessor scheduling problem, we can consider factors such as throughput, finishing time and processor utilization [2] [12].

Genetic algorithm is based on finishing time of a schedule. The finishing time of a schedule, S is defined as follows:

$$FT(S) = \underset{p_j}{Max}\ \ ftp\ (P_j)$$

Where, ftp (Pj) is the finishing time for the last module in processor P_j. To maximize the fitness function, we need to convert the finishing time into maximization form. This can be done by defining the fitness value of schedule, S, as follows:

$$C_{max} = FT(S)$$

Where, C_{max} is the maximum finishing time observed so far. Thus the optimal schedule would be the smallest finishing tie and a fitness value larger than the other schedules [1] [2] [6] [14] [16].

C. *Genetic Operators*

Function of genetic operators is to create new search nodes based on the current population of search nodes. By combining good structures of two search nodes, it may result in an even better one.

For multiprocessing scheduling problem, the genetic operators used must enforce the intra-processor precedence relations, as well as completeness and uniqueness of the modules in the schedule [1] [2] [3].

For multiprocessor scheduling, certain portions of the schedule may belong to the optimal schedule. By combining several of these optimal parts, we can find the optimal schedule efficiently.

For multiprocessing scheduling problem, the genetic operators used must enforce the intra-processor precedence relations, as well as completeness and uniqueness of the modules in the schedule [1] [2] [3].

Crossover:

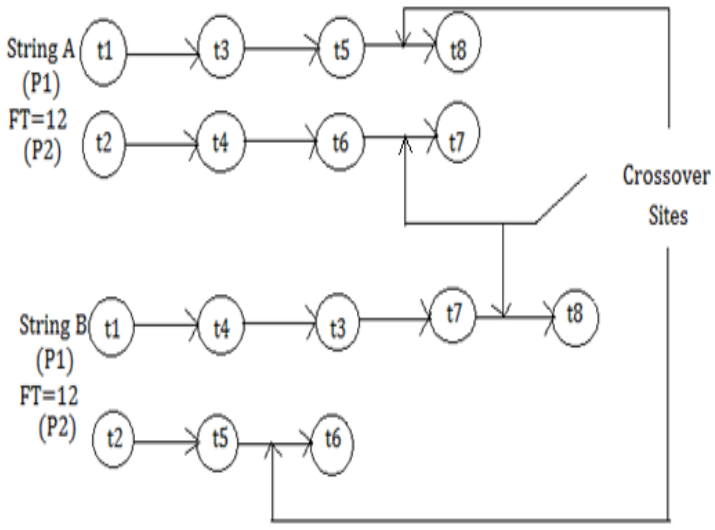

Fig 5: Two string for crossover operation

We can create new strings by exchanging portions of two strings by using following method:
1. Select sites where we can cut the lists into two halves
2. Exchange bottom halves of P_1 in string A and string B
3. Exchange bottom halves of P_2 in string A and string B.

For multiprocessor scheduling, we have to ensure that precedence relation is not violated and that the completeness and uniqueness of modules still holds after crossover [1] [2] [3] [6] [11] [17].

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

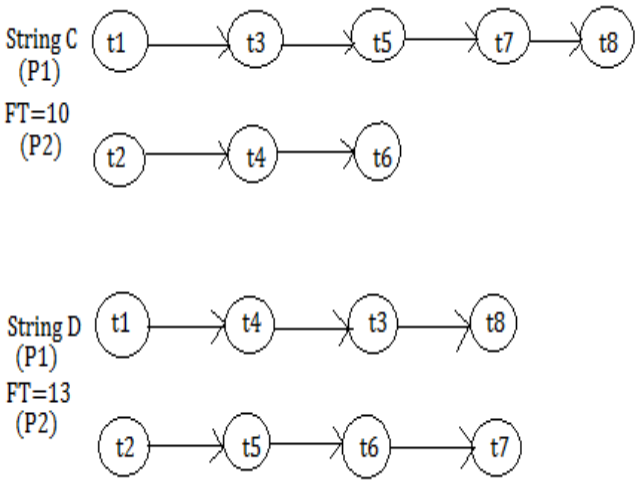

Fig 6: Two new strings generated after crossover operation

Reproduction:

Reproduction process forms a new population of strings by selecting string in the old population based on their fitness values. The selection criterion is that the strings with higher fitness value should have higher chance of surviving to next generation. Good strings have high fitness value and hence should be preserved into next generation [2] [6].

Mutation:

For multiprocessor scheduling problem, mutation is applied by randomly exchanging two modules with same height.

Algorithm using GA:

//Algorithm Find-Schedule
1. [Initialize]
2. Repeat steps 3 to 8 until algorithm is convergent
3. Compute fitness values for each string in the initial population
4. Perform Reproduction. Store string with highest fitness values in BEST_STRING.
5. Perform crossover
6. Perform mutation
7. Preserve the best string in BEST_STRING

Termination condition:

Best solution in the population obtained doesn't change after a specific number of generations. Here the algorithm is assumed to be convergent [1] [2].

D. An Efficient Multiprocessor Scheduling Algorithm Based On Genetic Algorithms

Typically, in previously described genetic algorithm, a biased roulette wheel is used to implement reproduction. Each string in the population occupies a slot size proportional to its fitness value. Random numbers are generated and used as an index into the roulette wheel to determine which string will be passed to the next generation. As the strings with higher fitness will have larger slot, they are more likely to be selected and passed onto the next generation.

We can make a slight modification to the basic reproduction operation by always passing the best string in the current generation to the next generation. This modification will increase the performance of the genetic algorithm. Algorithm is as follows:

Algorithm Reproduction

This algorithm performs the reproduction operation on a population of strings POP and generates a new population of strings NEWPOP.

1. [Initialize.] Let NPOP ← number of strings in POP.
2. [Construct the roulette wheel.] NSUM ← sum of all the fitness value of the strings in POP; form NSUM slots and assign string to the slots according to the fitness value of the string.
3. [Loop NPOP - I times.] Do step 4 NPOP - I times.
4. [Pick a string] Generate a random number between I and NSUM and use it to index into the slots to find the corresponding string; add this string to NEWPOP.
5. [Add the best string.] Add the string with the highest fitness value in POP to NEWPOP.

E. A Modified Multiprocessor Scheduling Algorithm Based On Memetic Algorithms

For this, a new technique called Simulated Annealing is used. It is a related global optimization technique which traverses the search space by testing random mutations on an individual solution. A mutation that increases fitness is always accepted. A mutation which lowers fitness is accepted probabilistically. SA here can be used within a standard GA algorithm, simply by starting with a relatively high rate of mutation, which decreases over time along a given schedule.

This combination of SA and GA itself is called memetic algorithm and it definitely improves the performance of scheduling algorithm. We can use the following algorithm with other genetic operators.

Algorithm modified-Find-Schedule
1. [Initialize]
2. Repeat steps 3 to 8 until algorithm is convergent
3. Compute fitness values for each string in the initial population.
4. Arrange the chromosomes according to decreasing order of their fitness values.
5. Discard 'p' lowest chromosomes.
6. Perform Reproduction on the remaining chromosomes. Store string with highest fitness values in BEST_STRING.
7. Perform crossover
8. Perform mutation
9. Preserve the best string in BEST_STRING

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 11, May 2014

In this, algorithm value of' 'p' depends on total number of chromosomes generated as the initial population and on the termination conditions.

IV. CONCLUSIONS

The problem of scheduling of modules to be executed on a multiprocessor system is one of the most challenging problems in parallel computing. Genetic algorithms are well adapted to multiprocessor scheduling problems.

Several genetic algorithms have been developed for multiprocessor module scheduling. The structure and restrictions placed upon the chromosomal representation significantly impact the complexity of genetic operators as well as the algorithm's potential for convergence to an optimal solution.

There are still some flaws in pure genetic algorithms and hence it is combined with simulated annealing to improve the overall performance of system. This combination itself is called memetic algorithm.

REFERENCES

[1] David E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning, Reading, Mass". Addison-Wesley, 1989.

[2] Edwin S. H. Hou, Ninvan Ansari and Hong Ren, "A Genetic Algorithm for Multiprocessor Scheduling", IEEE Transactions on Parallel and Distributed Systems. Vol. 5, No. 2, February 1994.

[3] Ricardo C. CorreA, Afonso Ferreira and Pascal Rebreyend, "Scheduling Multiprocessor Tasks with Genetic Algorithms", IEEE Transactions on Parallel and Distributed Systems, Vol. 10, No.8, August 1999.

[4] Annie S. Wu, Han Yu, Shiyuan Jin, Kuo-Chi Lin, and Guy Schiavone, "An incremental Genetic Algorithm Approach to Multiprocessor Scheduling", IEEE Transactions On Parallel And Distributed Systems, Vol. 15, No. 9, September 2004.

[5] Pai-Chou Wang, Willard Korfhage, "Process Scheduling Using Genetic Algorithms", IEEE, 1995.

[6] E. S. H. Hou, R. Hong, and N. Ansari, "Efficient Multiprocessor Scheduling Based On Genetic Algorithms", *IEEE, 1990.*

[7] Yi-Hsuan Lee and Cheng Chen, "A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems".

[8] Albert Y. Zomaya, Chris Ward, and Ben Macey, "Genetic Scheduling for Parallel Processor Systems: Comparative Studies and Performance Issues", IEEE Transactions On Parallel And Distributed Systems, Vol. 10, No. 8, August 1999.

[9] Albert Y. Zomaya, Senior Member, IEEE, Matthew Clements, And Stephan Olariu, "A Framework For Reinforcement-Based Scheduling In Parallel Processor Systems", IEEE Transactions On Parallel And Distributed Systems, Vol. 9, No. 3, March 1998.

[10] 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.

[11] Po-Jen Chuang and Chia-Hsin Wei, "An Efficient Optimization Technique for Task matching and Scheduling in Heterogeneous Computing Systems", IEEE proceedings of International Conference on Parallel and Distributed Systems 2002.

[12] Albert Y. Zomaya, And Yee-Hwei, "Observations On Using Genetic Algorithms For Dynamic Load- Balancing", IEEE Transactions On Parallel And Distributed Systems, Vol. 12, No. 9, September 2001.

[13] Sung-Ho Woo, Sung-Bong Yang, Shin-Dug Kim, and Tack-Don Han, "Task Scheduling in Distributed Computing Systems with a Genetic Algorithm", IEEE, 1997.

[14] Ricardo C. Correa, Afonso Ferreirat and Pascal Rebreyend, "Integrating list heuristics into genetic algorithms for multiprocessor scheduling", IEEE, 1996.

[15] Imtiaz Ahmad and Muhammad K. Dhodhi, "Multiprocessor Scheduling Using a Problem-Space Genetic Algorithm", Genetic Algorithms in Engineering Systems: Innovations and1 Applications 12-14 September 1995, Conference Publication No. 414,O IEE, 1995.

[16] Anna Swiecicka, Franciszek Seredynski, "Appling Cellular Automata in Multiprocessor Scheduling", IEEE proceedings of International Conference on Parallel Computing in Electrical Engineering 2002.

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

[18] Andrew J. Page and Thomas J. Naughton, "Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing".

[19] Gyoung Hwan Kim, And C. S. George Lee, "Genetic Reinforcement Learning Approach To The heterogeneous Machine Scheduling Problem", IEEE Transactions on Robotics and Automation, Vol. 14, No. 6, December 1998.

[20] K. M. Baumgarner and B. W. Wah, "Computer Scheduling Algorithms: Past Present and Future", First Workshop on Parallel Processing, National Tshing Hua University, Taiwan, December 1990.

[21] Amjed Mahmod, "A hybrid Genetic Algorithm for Task Scheduling in Multiprocessor Real time Systems".