International Journal of Computer Applications (0975 – 8887)
Volume 78 – No.4, September 2013

Optimization of Multiprocessor Scheduling using Genetic Algorithm

Poonam Panwar
Assistant Professor
Ambala College of Engineering & Applied Research, Ambala-133101

Shreya Chauhan
M.Tech Student
Ambala College of Engineering & Applied Research, Ambala-133101

ABSTRACT

Multiprocessor architectures are becoming more attractive for embedded systems, primarily because major processor manufacturers like Intel and AMD are designing cost effective processors even for personal computers and laptops. This makes such architectures very desirable for embedded system applications with high computational workloads, where additional, cost-effective processing capacity is often needed. This increased usage of multiprocessor attracted the researchers for multiprocessor scheduling problems. Multiprocessor scheduling is a NP hard problem. In this paper a Genetic Algorithm (GA) based multiprocessor scheduling algorithm is proposed whose implementation is simple and the obtained results are optimal for the studied set of problems.

Keywords

Multiprocessor Architecture, Multiprocessor Scheduling, NP Hard, Genetic Algorithm.

1. INTRODUCTION

Real-time multiprocessor systems are now very commonly used for a number of applications in our life. There are a number of designs available for multiprocessor architectures like single-chip architectures, multiprocessor with a modest number of processors and large-scale signal-processing systems such as synthetic-aperture radar systems etc. In a single processor system the problem of ensuring that deadline constraints are met has been widely studied and the effective scheduling algorithms that take into account the many complexities that arise in real systems like synchronization costs, system overheads, etc. are well understood. In contrast to single processor systems researchers are just beginning to understand the trade-offs that exist in multiprocessor systems. Scheduling approaches for Multiprocessor systems can be classified into two categories: global scheduling and partitioning. In global scheduling, all eligible tasks are stored in a single priority-ordered queue then the global scheduler selects for execution the highest priority tasks from this queue. Unfortunately, using this approach with optimal single processor scheduling algorithms, such as the rate-monotonic (RM) and earliest-deadline-first (EDF) algorithms may result in arbitrarily low processor utilization in multiprocessor systems [1-2]. However, recent research on proportionate fair (Pfair) scheduling has shown considerable promise in that and it has produced the only known optimal method for scheduling periodic tasks on multiprocessors [3-4].

In partitioning, each task is assigned to a single processor, on which each of its jobs will execute, and processors are scheduled independently. The main advantage of partitioning approaches is that they reduce a multiprocessor scheduling problem to a set of single processor or uniprocessor ones. Unfortunately, partitioning has two negative consequences. First, finding an optimal assignment of tasks to processors is a bin-packing problem, which is NP-hard in the strong sense. Thus, tasks are usually partitioned using non-optimal heuristics. Second task systems exist that are schedulable if and only if tasks are not partitioned. Still, partitioning approaches are widely used by system designers [5-8].

In addition to the above approaches the approach considered in proposed work is a new “middle” level approach in which each job is assigned to a single processor, while a task is allowed to migrate. In other words, inter-processor task migration is permitted only at job boundaries. It is believed that migration is eschewed in the design of multiprocessor real-time systems because its true cost in terms of the final system produced is not well understood. As a step towards understanding this cost, a new taxonomy that ranks scheduling schemes is the complexity of priority schemes. According to this dimension the scheduling disciplines are categorized according to the task priorities. The task priorities can be static, dynamic but fixed within a job or fully dynamic. Common examples of each type include RM, EDF and least-laxity-first (LLF) [9-12] scheduling algorithms.

2. PROPOSED APPROACH

The classical multiprocessor scheduling techniques are having number of pitfalls like they takes more execution time, the basic solution obtained without swarm intelligence is not efficient in terms of the turnaround time and optimal results. Due to this there is a requirement of optimization of multiprocessor scheduling. Genetic algorithm (GA) is a technique that is applied to a number of optimization problems to obtained optimal solutions. So in proposed a GA based approach is proposed for optimization of multiprocessor scheduling. A Genetic Algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [13]. In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached [14-19]. In proposed study the drawbacks and shortcomings in the classical multiprocessor scheduling are investigated and a novel technique for the simulation of genetic algorithm based multiprocessor scheduling is proposed to optimize execution time. Matlab GA Tool is used for proposed approach with following GA parameters:

- Initial Population size: popSize=10
- Maximum number of iterations: maxGen=1000
- Crossover probability=0.8
- Mutation probability=0.8
- Terminating condition: 100 generations with same fitness.
- Total number of trials= 10 for each problem.

The algorithm to get optimal execution time includes following steps:

1. Design computation cost matrix $\mathrm{ccm}[m,n]$ where m is the total number of tasks $(t1, t2, \dots, tm)$ and n is the number of processors (i.e. $p1, p2, \dots, pn$). Each entry in matrix denotes the execution time on specified processor.
2. Design communication matrix $\mathrm{cc}[m,m]$ where each entry denotes the weight of node from parent to child. The computation cost is equal to 0 if child is executed on the same processor on which parent has executed.
3. Assign priorities to tasks i.e. static, dynamic but fixed or fully dynamic.
4. Calculate EST (Earlist Start Time) and EFT(Earliest Finish Time) as specified below:

   Set $\mathrm{EST} = 0$ and EFT=0 for task 1 on each processor, set process avail time for processors $p1 = p2 = \cdots = pn = 0$. Assign that processor to task1 on which execution time (EFT) is minimum and set the value of corresponding processor as:

   $$p_i = \mathrm{EFT}(t_1) \qquad \text{where } i = 1,2,\dots m \qquad (1)$$

5. For t_2 to t_n repeat following:

   $$\mathrm{EFT}(t_i, p_j) = \max\big(\mathrm{EFT}(t_{parent}) + \mathrm{cc}[i,j], \text{processor avail time} + \mathrm{cc}[i,j]\big) \qquad (2)$$

   where t_{parent} is the parent node of t_i and $i = 2,3,\dots m$ and $j = 1,2,\dots,n$.
6. Fitness function = $\min\big(\mathrm{EFT}(t_i)\big)$ for chromosome 1 to k where k is the total number of chromosomes in a generation.
7. Calculate total execution time of as $\max\,(\mathrm{EFT}(t_i))$.

3. IMPLEMENTATION OF PROPOSED ALGORITHM

A problem from existing literature [14] is selected from the literature. The application consists of 10 tasks. It starts execution from first task and end at the execution of last task. The architecture of the problem is shown in figure1. We have implemented the multiprocessor scheduling with and without genetic algorithm on different cases of inputs. The corresponding task graphs and Gantt charts has been generated to analyze the tasks.

CASE – I

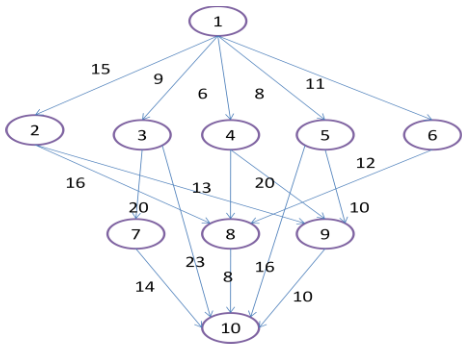

Fig 1: Directed acyclic graph of case1.

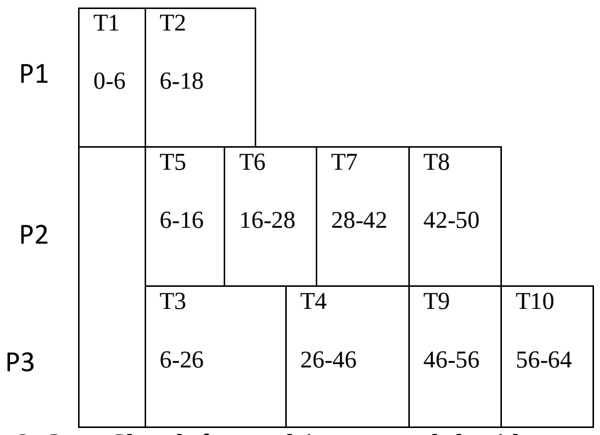

Fig 2: Gantt Chart before applying proposed algorithm on case I (Total Execution time= 64).

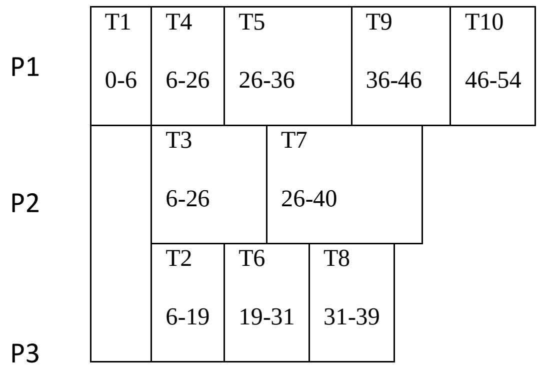

Fig 3: Gantt Chart after applying genetic algorithm on case I (Total execution time= 54)

CASE II:

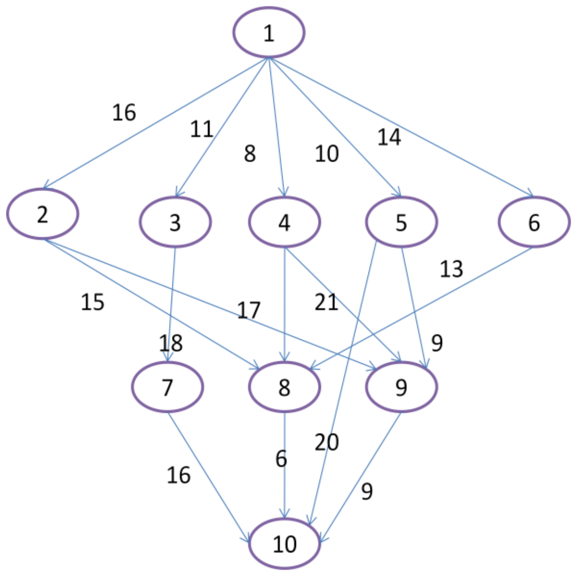

Fig 4: Directed acyclic graph of case2.

P1	T1 0-8	T2 8-23			
P2		T5 8-17	T6 17-30	T7 30-46	T8 46-52
P3		T3 8-26	T4 26-47	T9 47-56	T10 56-62

Fig 5: Gantt Chart before applying proposed algorithm on case II (Total Execution time= 62).

P1	T1 0-8	T4 8-29	T5 29-38	T9 38-47	T10 47-53
P2		T3 8-26	T7 26-42		
P3		T2 8-23	T6 23-36	T8 36-42	

Fig 6: Gantt Chart after applying genetic algorithm on case II (Total execution time= 53)

CASE III

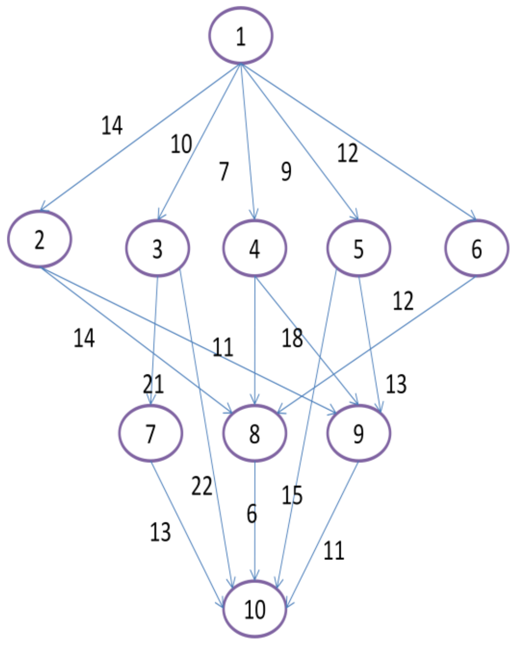

Fig 7: Directed acyclic graph of case3.

P1	T1 0-7	T2 7-18			
P2		T5 7-20	T6 20-32	T7 32-45	T8 45-51
P3		T3 7-28	T4 28-46	T9 46-57	T10 57-63

Fig 8: Gantt Chart before applying proposed algorithm on case III (Total Execution time= 63).

P1	T1 0-7	T4 7-25	T5 25-38	T9 38-49	T10 49-55
P2		T3 7-28	T7 28-41		
P3		T2 7-18	T6 18-30	T8 30-36	

Fig. 9: Gantt Chart after applying genetic algorithm on case III (Total execution time= 55)

As shown in Fig 2 to Fig 9 it can be concluded that the proposed approach is generating efficient results in terms of the optimal solution when executed using genetic algorithm. The proposed technique is efficient also in terms of the execution and turnaround time despite of the number of iterations. But it also has the limitation that it further needs enhancement using assorted meta-heuristics and the approach may give better results by combining simulated annealing with the proposed operators.

4. CONCLUSION AND FUTURE WORK

The existing problems in the multiprocessor scheduling [20-22] have been removed using genetic algorithm and optimal results has been obtained. The further work in this area can be improved by using the other meta-heuristics including ant colony optimization, simulated annealing, honeybee algorithm. These algorithms are very prominent in terms of solving the combinatorial optimization problems. The multiprocessor scheduling algorithm can be passed with assorted swarm intelligence techniques to get the multiple results and from which the optimal result should be obtained.

5. REFERENCES

[1] Anderson J. and Srinivasan A. June 2001. Mixed Pfair/ ERfair scheduling of asynchronous periodic tasks in proceedings of the 13th Euromicro Conference on Real-time Systems, 76–85.

[2] Moir M. and Ramamurthy S. 1999. Pfair scheduling of fixed and migrating periodic tasks on multiple resources in proceedings of the 20th IEEE Real-time Systems Symposium, 294–303.

[3] Srinivasan A., Holman P., Anderson J., and Baruah S. 2003. The case for fair multiprocessor scheduling in proceedings of the 11th International Workshop on Parallel and Distributed Real-time Systems.

[4] Baruah S., Cohen N., Plaxton C.G., and Varvel. D. 1996. Proportionate progress: A notion of fairness in resource allocation. In Algorithmica, 15:600–625.

[5] Andersson B., Baruah S., and Jansson J. 2001. Static-priority scheduling on multiprocessors in proceedings of the 22nd IEEE Real-time Systems Symposium, 193–202.

[6] Srinivasan A. and Baruah S. 2002. Deadline-based scheduling of periodic task systems on multiprocessors in Information Processing Letters, 84(2):93–98.

[7] Topcuoglu H., Hariri S. 2002. Performance Effective and Low Complexity Task Scheduling for Heterogenous Computing in IEEE Transactions on Parallel and Distributed Systems, Vol-13 No.3.

[8] Anderson J. and Srinivasan A. 2000. Early-release fair scheduling in Proc. of the 12th Euromicro Conference on Real-Time Systems, 35–43.

[9] Mitchell M. 1995. Genetic Algorithms: An Overview in An Introduction to Genetic Algorithms, Chapter 1. MIT Press.

[10] Konfrst Z. 2004. Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives in18th International Parallel and Distributed Processing.

[11] Baskiyar S. and SaiRanga P.C. 2003. Scheduling Directed A-cyclic Task Graphs on Heterogeneous Network of Workstations to Minimize Schedule Length in Proceedings of the IEEE International Conference on Parallel Processing Workshops (ICPPW"03).

[12] Blythe J., Jain S., Deelman E., Gil Y., Vahi K, Mandal A, and Kennedy K. 2005. Task scheduling strategies for workflow based applications in grids in CCGRID, 759-767.

[13] Gen M, Cheng R. 2000. Genetic algorithm and engineering optimization. NewYork:Wiley.

[14] ESH, Ansari N, Hong R. 1994. A genetic algorithm for multiprocessor scheduling in IEEE Transactions on Parallel and Distributed Systems; 5(2):113–20.

[15] Hwang R, Gen M. 2004. Multiprocessor scheduling using genetic algorithm with priority-based coding in Proceedings of IEEJ conference on electronics, information and systems.

[16] Reakook H., Mitsuo G. and Hiroshi K.. 2008. A comparison of multiprocessor task scheduling algorithms with communication costs in Computers and Operations Research; 35, 976-993.

[17] Ilavarasan E., Thambidurai P. and Mahilmannan. 2005. Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System in proceedings of the 4th International Symposium on Parallel and Distributed Computing.

[18] Kwok Y.K. and Ahmad I. 1999. Static Scheduling algorithms for allocating directed task graphs to multiprocessors in ACM Comput. Surv. 31(4), 406-471.

[19] EI-Rewini H., Lewis T.G., and Ali H.H. 1994. Task Scheduling in parallel and distributed Systems in Prentice Hall, Englewood Cliffs.

[20] Topcuoglu H., Hariri S. and Min-You Wu. 2002. Performance-Effective and Low-Complexity Task Scheduling for Heterogenous Computing in IEEE transactions on Parallel and Distributed Systems, Vol. 13, no. 3.

[21] Tsujimura Y, Gen M. 1995. Genetic algorithms for solving multiprocessor scheduling problems. In: Simulated evolution and learning. Heidelberg: Springer; 106–15.

[22] Yang J. XiaochuanMa, Chaohuan Hou and Zheng Yao. 2008. A Static Multiprocessor Scheduling Algorithm for Arbitrary Directed Task Graphs in Uncertain Environments in Springer-Verlag Berlin Heidelberg.