Journal of Engineering Sciences, Assiut University, Vol. 37, No. 3, pp. 691-698 ,May 2009.

SCHEDULING REAL-TIME TASKS IN MULTIPROCESSOR SYSTEMS USING GENETIC ALGORITHMS

E. M. Saad[1], H. A. Keshk[1], M. A. Saleh[1], and A.A.Hamam[2]
[1] *Faculty of Engineering Helwan University, Helwan, Egypt*
[2] *Thebes Academy, Giza, Egypt*

(Received March 23, 2009 Accepted May12, 2009)

Multiprocessors have been employed as a powerful computing means for executing real-time tasks, especially where a uniprocessor system would not be sufficient to execute all the tasks. This paper investigates dynamic scheduling algorithm for real-time tasks in a multiprocessor systems to obtain a feasible solution using genetic algorithms combined with earliest deadline first (EDF) and shortest computation time first (SCTF). A comparative study of the results obtained from simulations shows that genetic algorithm can be used to schedule tasks to meet their deadlines time , in addition to obtain high processor utilization..

KEYWORDS *– Task Graph, Optimization, Real-time system, genetic algorithms, multiprocessor scheduling*

1. INTRODUCTION

Real-time systems are systems in which the time at which the result is produced is as important as the logical correctness of the result [1-3]. Real-time applications span a large range of activities which include production automation, nuclear plant supervision, command and control systems, and robotics and banking transactions [2].

Scheduling is an important aspect in real-time systems to ensure soft/hard timing constraints. Scheduling tasks involves the allocating of resources to tasks, to satisfy certain performance needs [1]. In real-time applications, real-time tasks are the basic executable entities that are scheduled [2]. The tasks may be periodic or a periodic and may have soft or hard real-time constraints. Scheduling task set consists of planning the order of execution of task request so that the timing constraints are met.

Multiprocessors have emerged as a powerful computing means for running real-time applications, especially where a uniprocessor system would not be sufficient enough to execute all the tasks by their deadlines [4].

Real-time systems make use of scheduling algorithms to maximize the number of real-time tasks that can be processed without violating timing constraints [5]. A scheduling algorithm provides a schedule for a task set that assigns tasks to processors and provides an ordered list of tasks. The schedule is said to be feasible if the timing constraints of all the tasks are met [2].

In multiprocessor real-time systems static algorithms are used to schedule periodic tasks whose characteristics are known priori. Scheduling of periodic tasks whose characteristics are not known a priori requires dynamic scheduling algorithms [5]. Dynamic scheduling can be either centralized or distributed. In a distributed

dynamic scheduling scheme, each processor has its own local scheduler that determines whether it can satisfy the requirements of the incoming task. In a centralized dynamic scheduling scheme, there is a central processor called the scheduler which determines which processor the task should be allocated for execution [5].

Various heuristics approaches have been widely used for scheduling. The use of genetic algorithm (GA) for real-time task scheduling is to be studied extensively. GAs may be seen attractive to real-time application designer as it relieves the designer from knowing how to assess a given solution.

In this paper, we present a scheduling algorithm using genetic algorithms combined with traditional scheduling heuristic to generate a feasible schedule based on the work done by Saad *et al*., [6]. The paper aims in meeting deadlines and provides a comparative study of merging heuristics methods such as Earliest Deadline First (EDF) and Shortest Computation Time First (SCTF) separately with genetic algorithms.

The paper is organized as follows; section 2 reviews the related work in the area of scheduling real-time tasks. Section 3 describes task and scheduler models. Section 4 discussed proposes scheduling algorithm. In section 5, simulation results are presented. Finally in section 6 conclusion and summary are presented.

2- RELATED WORK

The problem of scheduling real-time tasks on multiprocessor systems using genetic algorithms has been studied extensively and a number of algorithms have been proposed [7-11].

Wu *et al*., [7], developed a genetic algorithm (GA) approach for the problem of task scheduling for multiprocessor systems. Key features of this approach include a flexible, adaptive problem representation and an incremental fitness function. The advantages of this algorithm are that it is simple to use, require minimal problem specific information and is able to effectively adapt in dynamically changing environment. The primary disadvantage of this algorithm is that it has a long execution time.

Kamiura *et al*., [8], developed a parallel genetic algorithm for multi-objective optimization problems (MOGADES) which can derive widespread Pareto optimal solutions.

Page *et al*., [9], developed a scheduling algorithm to schedule heterogeneous processors in a distributed computing system. This algorithm provides efficiently schedules and adapts to varying resources availability. Also, it consistently uses processors more efficiently than the current state-of-the-art GA algorithms for the same problems. The most important drawback of this algorithm that is not tested under the real-world condition and the efficiency of the algorithm for time critical applications have not been studied.

Zomaya, *et al*., [10], presented a scheduling algorithm based on genetic algorithms, to efficiently solve the scheduling problem without the need to apply any restricted assumptions that are problem-specific such is the case when using heuristics.

Tripathi, *et al*., [11], developed a method based on genetic algorithms to allocating multiple tasks on heterogeneous distributed computing system taking into account the execution and communication costs.

Golub, *et al*., [12], developed an efficient genetic algorithm for scheduling precedence constrained task graphs without taking into account the communication cost. This algorithm based on the task priorities; it select the task with highest priority first. If more than one task has the same priority the task is selected randomly.

3. TASK AND SCHEDULER MODELS

The real-time system is assumed to consist of m, identical processors for the execution of time it scheduled tasks where m>1. They are assumed to be connected through a fully connected network. The scheduler may assign a task to any one of the processors. Each task T_i in the task set is considered to be aperiodic, independent and non-preemptive. Each task is characterized by A_i: arrival time, C_i : computation time, D_i : deadline [6].

The scheduler determines the scheduled start time $st(T_i)$ and the finish time $ft(T_i)$ of a task. The task T_i meets its deadline if $A_i \leq st(T_i) \leq D_i - C_i$ and $A_i + C_i$ $ft(T_i) \leq D_i$. That is , the tasks are scheduled to start after they arrive and finish execution before their deadlines [6].

We considered the incoming tasks are held in the task queue and passed on to the scheduler for scheduling of tasks. The central scheduler allocates the incoming tasks to the other processors in the system. The processor executes tasks in the order they arrive in the dispatch queue. The scheduler works in parallel with the processors. The scheduler deals with the newly arriving tasks and updates the queue while the processors execute the tasks assigned to them. A feasible schedule is determined by the scheduler based on the computation time of tasks satisfying their timing constraints.

The scheduler model showing the parallel execution of the scheduler and processors is shown in the following figure.

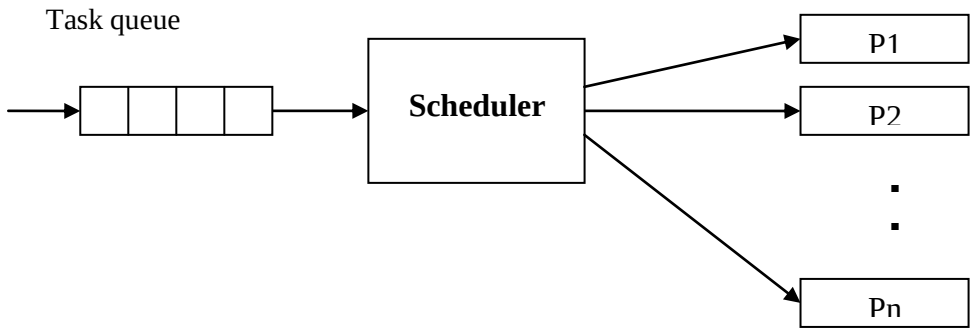


Figure (1): The scheduler model

4. PROPOSED SCHEDULING ALGORITHM

Before presenting our scheduling algorithm, we discuss genetic operators and chromosome syntax used for representation a task scheduling problem.

4.1. Chromosome Syntax

We have three basic elements to choose among them for chromosome representation for a task scheduling problem. The first is the list of tasks to be scheduled, the second is the order in which these tasks should be executed on a given set of processors and the third is the list of processors which tasks should be assigned to. So, the chromosomes representation for a task scheduling problem is one in which gene is a pair of decimal values (T_i, P_j) indicates that the task T_i is assigned to processor P_j . The position of genes in a chromosome specifies the order in which tasks should be executed.

ILL USTRATIVE EXAMPLE

The following chromosome syntax shows that the task 1 and task 3 should be executed on processor 1 and task 2 on processor 3. It also indicates that task 3 should be executed first, followed by task 2, and task one.

1	2	3
(3,1)	(2,3)	(1,1)

4.2. Genetic Operators

The crossover operator randomly selects two chromosomes from the population and swaps second part of each gene after a randomly selected point. This is equivalent to assigning a subset of tasks to different processors.

In mutation operator we use the operation called inversion in which we randomly select two points in chromosome and reverse the order of the genes between these two points. This operation is followed by another operation which randomly selects a chromosome and changes a randomly selected gene (T_i, P_i) to (T_i, P_j) over all processors. This makes use of the heuristic that a task should be assigned to a processor where it has earliest start time.

4.3 Fitness Function

For task scheduling problem we used a simple evaluation function in which the fitness value of a chromosome is determined by the number of tasks that meet their deadlines in a chromosome.

4.4. Proposed GA Algorithm

Initially a task queue is generated with tasks having the flowing characteristics namely, arrival time, computation time and deadline time. The tasks are ordered so that the task with earliest deadline can be considered first for scheduling. The algorithm considers a set of tasks from the sorted list to generate an initial population. Each chromosome in the initial population is generated by assigning each task in the task set to a randomly

selected processor and the pair (Task, processor) is inserted in a randomly selected unoccupied locus of the chromosome. The size of chromosome depends on the number of tasks selected from the sorted list. The tasks in each chromosome are then sorted based on their deadline. This is done because the chromosome representation also gives the order in which the tasks are executed. The fitness evaluation of the chromosome in the population is then performed to determine the number of tasks in each chromosome that meet their deadlines. The chromosomes are then sorted based on the fitness value in a descending order.

GA operators are then applied to population of chromosome until reaching the fitness value. In each iteration, the tasks in the chromosome are sorted based on their deadline and the evaluation and sorting of chromosome based on fitness value is performed. After satisfying the fitness value, the best schedule for the set of tasks is obtained.

The steps of proposed algorithm are as follows:

1. generate a task queue
2. Sort the task in the increasing order of their deadlines.
3. Select a suitable number of tasks for a fixed chromosome size.
4. Generate chromosomes for the population.
5. Sort the genes in each chromosomes based on deadlines.
6. Determine the fitness value of each chromosome in the population.
7. Sort the chromosome with in the population depending on fitness value.
8. Apply GA operators for a number of iterations:

8.1 Sort the genes in each chromosome based on deadline.

8.2 Determine the fitness value of each chromosome in the population.

8.3 Sort the chromosome within the population depending on the fitness value.

9. Choose the best chromosome. The tasks that are found infeasible are removed from the chromosome so that they are not reconsidered for scheduling.

5. SIMULATION RESULTS

To study the effectiveness of the proposed task scheduling algorithm, we conducted extensive simulation studies. The performance of the algorithm is measured by the number of executed tasks.

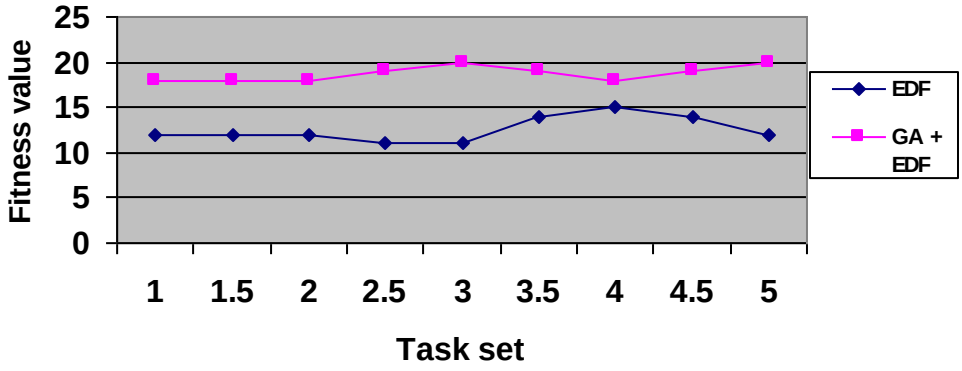
The simulation study considers the assigning of a set of tasks to a number of processors. The tasks are generated and put in the task queues of different length 100,200, and 400 tasks. The computation time for each task has been chosen randomly between 30 and 60 time units. The value of a deadline of a task is equal to $A_i + C_i +$ variable value between (60, 70) time units [6]. This ensures that the computation time is always less than deadline. The arrival time of the tasks has exponential distribution with arrival rate λ=0.5. The number of processors, m is taken from the optimal situation reported in [6].

The value of the number of iteration has been assumed 20. The chromosome size has been assumed equal to the number of tasks considered at a time for scheduling. The value of chromosome size has been varied between 20 and 40. As mentioned before the fitness value determines the number of tasks in the chromosome that can meet their deadlines. Hence, here for chromosome size 20 the maximum fitness value that can be obtained is 20.

The population size for the algorithm has been assumed to be 30. That is to say those 30 chromosomes have been considered at a time for the application of GA operators. For an initial evaluation, the fitness value was calculated by assigning tasks based on EDF. For a task queue of 100 tasks, it divided into task sets of 20 each. The processor was chosen between processor 1 and processor 10.

The fitness value obtained for each task set is shown in Figure (2). The graph shows that the maximum number of tasks that met their deadline is 15 when considering 20 tasks for scheduling. The majority of the task sets gave a fitness value 12 (that is, 12 tasks out of 20 met their deadline). Our proposed algorithm was used to schedule the same task sets. The algorithm is combined with the heuristic EDF. The graph showing the fitness value of tasks was obtained which has better performance. Thus it could be seen that the percentage of tasks that are feasible is 95% and above compared to 75 % for EDF only.

Fig. (2): Task feasibility for EDF and combined GA



Also, the comparative study for the different task queues length 100, 200 and 400 for two different chromosome sizes 20 and 40 are examined. The results were recorded for two cases that is, using proposed algorithm combined with two heuristics SCTF and EDF. Figure (3) shows that for all the cases the number of tasks that were feasible was 90% and above for both EDF and SCTF. Also, it could be seen that the use of SCTF gave better fitness values compared to EDF. This shows that the heuristic SCTF is a better option for combining with GA.

Fig. (3) Fitness values for chromosome sizes 20 and 40

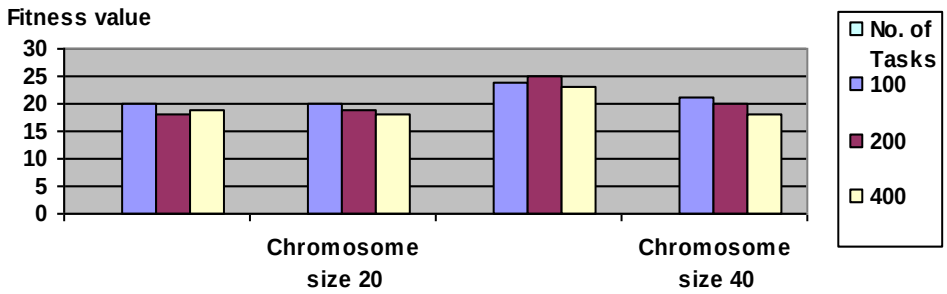


Table (1) shows the total time units taken to get the optimal solution by using our proposed algorithm and the algorithm in [12] for the different task queues length when chromosome size equal to 20. It is clear that the use of GA combining with SCTF and EDF give better results than [12].

Table (1); Total time units to get optimal solution

Task queue	GA + priorities	GA + SCTF	GA+ EDF
100	120	118	119
200	243	240	242
300	365	361	362
400	486	479	481

From the above results it could be said that traditional scheduling heuristics methods could be combined with GA to scheduled real-time tasks if scheduling time used by GA is reduced by some efficient methods.

6. CONCLUSION

Scheduling is an important topic that is applicable in a wide variety of domains. Generally, scheduling problems are NP-hard and there are no general algorithms that can guarantee an optimal solution. This is the same as in the case of scheduling real-time tasks as well. This paper has discussed that GA combining with traditional heuristics could be used to obtain feasible solutions. It is noted that GA combining with SCTF gave better performance as compared to the EDF. Also, the algorithm is successful in obtaining feasible solutions however; the implementation of the GA algorithm is quite costly.

REFERENCES

[1] K. Ramamritham, and J.A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems", Proceedings of IEEE, Vol. 82, No. 1, PP. 55-67, 1994.

[2] A. Mohammadi and S.G. Akl, "Scheduling algorithms for real-time systems", School of Computing, Queen's University, Kingston, Ontario Canada, july 15, 2005.

[3] L. Sha, T. Abd El Zaher, A. Karl- Erik, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real-time scheduling theory : A historical perspective", real-time systems, 28, 101-155, 2004.

[4] K. Ramamritham, J. A. Stankovic and S, Perng-Pei, "Efficient scheduling algorithms for real-time multiprocessor system", IEEE Transactions on Parallel and distributed Systems, Vol. 1, No. 2, April 1990.

[5] G. Manimaran, C. Sivaram Murthy, "An efficient dynamic scheduling algorithm for multiprocessor real-time systems", IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No. 3, PP. 312-319, 1998.

[6] E. M. Saad, H. A. Heshk, M. A. Saleh, and A. A. Hamam, "Scheduling hard real-time tasks with precedence constraints on multiprocessor systems", JES, Assiut University, Vol. 35, No. 6, PP. 1443-1453, November 2007.

[7] A. S. Wu, H. Yu, S. Jin, K. C. Lin, and G. Schiavene, "An incremental genetic algorithm approach to multiprocessor scheduling", IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 9, Sept. 2004, PP. 824-834.

[8] J. Kamiura, T. Hiroyasu, M. Miki, and S. Watanabe, "MOGADES: Multi-Objective genetic algorithm with distributed environment scheme", Computational Intelligence and Applications, Proceedings of the 2nd International Workshop on Intelligent System Design and Applications: ISDA-02, PP. 143 – 148, 2002.

[9] A. J. Page, and T. J. Naughton, "Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing", The proceedings of the 19th International parallel & Distributed processing Symposium, Denver, USA, IEEE Computer Society, 2005.

[10] A.Y Zomaya, C. Waad, and B. Macey, "genetic scheduling for parallel processing systems: Comperative studies and performance issues", IEEE Transactions on Parallel and Distributed Systems, vol. 10, No.8, PP. 799 – 812, Aug. 1999.

[11] A. K. Tripathi, B.k. Kumer and N. Kumar,"A GA based multiple task allocation considering load", International Journal of High Speed Computing, Vol. 11, No. 4, PP. 209 – 214, 2000.

[12] M. Golub, and S. Kasapovic, "Scheduling multiprocessor tasks with genetic algorithms", proceedings of the 23rd International Conference ITI 2001, Puls, 2001.

جدولة المهام للنظم متعددة المعالجات في الزمن الفعلي باستخدام الخوارزميات الجينية

تعتبر المعالجات المتعددة وسيلة قوية و فعالة لتنفيذ المهام في الزمن الفعلي خاصة أن النظم وحيدة المعالج تكون غير كافية لتفيذ كل المهام. في هذا البحث تم استخام تقتية الخوارزميات الجينية لجدولة المهام للنظم متعددة المعالجات في الزمن الفعلي مدمجا مع الطرق التقليدية مثل الوقت المبكر الذي لابد أن تنتهي فيه المهمة أولا و اقل وقت تنفيذ أولا. دراسة المقارنة للنتائج التي تم الحصول عليها من خلال الدراسة التجريبية توضح فاعلية الخوارزميات الجينية لجدولة المهام بالاضافة إلي الكفاء العالية للمعالجات.