Adaptive Scheduling Algorithm for Real-Time Operating System

Ketan Kotecha and Apurva Shah

Abstract— **EDF (Earliest Deadline First) has been proved to be optimal scheduling algorithm for single processor real-time operating systems when the systems are preemptive and underloaded. The limitation of this algorithm is, its performance decreases exponentially when system becomes slightly overloaded. Authors have already proved ability of ACO (Ant Colony Optimization) based scheduling algorithm for real-time operating system which is optimal during underloaded condition and it gives outstanding results in overloaded condition. The limitation of this algorithm is, it takes more time for execution compared to EDF algorithm.**

In this paper, an adaptive scheduling algorithm is proposed which is combination of both of these algorithms. Basically the new algorithm uses EDF algorithm but when the system becomes overloaded, it will switch to ACO based scheduling algorithm. Again, when the overload disappears, the system will switch to EDF algorithm. Therefore, the proposed algorithm takes the advantages of both algorithms and overcomes the limitations of each other.

The proposed algorithm along with EDF algorithm and ACO based scheduling algorithm, is simulated for real-time system and the results are obtained. The performance is measured in terms of Success Ratio and Effective CPU Utilization. Execution Time taken by each scheduling algorithm is also measured. From analysis and experiments it reveals that the proposed algorithm is fast as well as very efficient in both underloaded and overloaded conditions.

Keywords: Real-Time Operating Systems, Scheduling Algorithm, ACO Algorithms , EDF

I. Introduction

Real-time systems are defined as those systems in which the correctness of the system does not depend only on the logical results of computations but also on the time at which the results are produced[11]. Real-time systems have well defined, fixed time constraints i.e., processing must be done within the defined constraints otherwise the system will fail. There are two main types of real-time systems: Hard Real-Time System and Soft Real-Time System. Hard Real-Time System requires that absolute deadlines must be met otherwise catastrophic situation may arise whereas in Soft Real-Time System, missing an occasional deadline is undesirable, but nevertheless tolerable.

The objective of a real-time task scheduler is to guarantee the deadline of tasks in the system as much as possible when we consider soft real-time system. To achieve this goal, vast researches on real-time task scheduling have been conducted. Mostly all the real-time systems in existence use preemption and multitasking. Real-time scheduling techniques can be broadly divided into two categories: Static and Dynamic.

Static algorithms assign priorities at design time. All assigned priorities remain constant for the lifetime of a task. Dynamic algorithms assign priorities at runtime, based on execution parameters of tasks. Dynamic scheduling can be either with static priority or dynamic priority. RM(Rate Monolithic) and DM(Deadline Monolithic) are examples of dynamic scheduling with static priority[1]. EDF(Earliest Deadline First) and LST (Least Slack Time First) are examples of dynamic scheduling with dynamic priority[8]. EDF and LST algorithms are optimal under the condition that the jobs are preemptive, there is only one processor and the processor is not overloaded [2][3]. But the limitation of these algorithms is, their performance decreases exponentially if the system becomes slightly overloaded [4].

ACO is a branch of Swarm Intelligence. The advantages of ant-based systems are inherent parallelism, robustness & scalability along with simplicity of individual agent[5][6]. Authors have already proved that ACO based scheduling algorithm performs well during overloaded condition[7]. This algorithm has been already proved to be optimal when the system is underloaded. But the limitation of this algorithm is that it takes more time for execution compared to EDF algorithm.

The whole paper is organized as follows: In Section 2, the proposed algorithm is explained and discussed. Section 3 contains simulation method and performance measuring parameters. Section 4 contains the results obtained and the paper ends with a brief conclusion in Section 5.

II. The Adaptive Algorithm

A. System and Task Model

We call each unit of work that is scheduled and executed by the system as a job and a set of related jobs, which jointly provide some system function, is a task[8]. All the tasks are assumed to be periodic. The system knows about arrival time, period, required execution time and deadline of the task in priori. There are no precedence constraints on the task; they can run in any order relative to each other as long as their deadlines are met. A task is ready to execute as it arrives in the system. We have assumed that the system is not having resource contention problem. The task set is assumed to be preemptive. We have also assumed that preemption and the scheduling algorithm incurs no overhead.

In soft real-time systems, each task has a positive value. The goal of the system is to obtain as much value as possible. If a task succeeds, then the system acquires its value. If a task fails, then the system gains less value from the task[12]. In

Ketan Kotecha is the Principal of G H Patel College of Engg. & Tech., Charutar Vidya Mandal, Vallabh Vidyanagar - 388 120, INDIA (email: drketankotecha@gcet.ac.in).

Apurva Shah is a Research Scholar at Sardar Patel University, Vallabh Vidyanagar - 388 120, INDIA (email: ams_gcet@yahoo.co.in).

978-1-4244-1823-7/08/$25.00

a special case of soft real-time systems, called a firm real-time system, there is no value for a task that has missed its deadline, but there is no catastrophe either[13]. Here, we propose an algorithm that applies to firm real-time system. The value of the task has been taken same as its execution time required.

B. The Algorithm

The Adaptive algorithm is combination of two scheduling algorithms: EDF algorithm and ACO based Scheduling algorithm.

1) EDF Algorithm:

   The priority of each task is decided based on the value of its deadline. The task with nearest deadline is given highest priority and it is selected for execution. This algorithm is simple and proved to be optimal when the system is preemptive, underloaded and there is only one processor.

2) ACO Based Scheduling Algorithm:

   The ACO algorithms are computational models inspired by the collective foraging behavior of ants [9]. Each ant is an autonomous agent that constructs a path. There might be one or more ants concurrently active at the same time. Ants do not need synchronization. Forward ant moves to the good-looking neighbor from the current node, probabilistically. A probabilistic choice is biased by Pheromone trails previously deposited and heuristic function. Without heuristics information, the algorithm tends to converge towards initial random solution. In backward mode, ants lay down the pheromone. Pheromone intensity of all the paths decreases with time, called pheromone evaporation. It helps in unlearning poor quality solution [9].
   Pseudo-code of the ACO based scheduling algorithm is given as per following [7]:

   ACO_based_Scheduling
   {
   - Construct the tour of different ants
   - Analyze the results of ants' journeys
   - Update the value of Pheromone
   - Find probability of each task and select the task for execution

   }
   In ACO based scheduling algorithm, each schedulable task is considered as a node and all the ants will start their journey from different nodes. Number of ants are taken same as number of schedulable tasks at that time. The ants will traverse depending on the value of pheromone and some heuristic function. Pheromone value will be updated on each node depending on the performance of the journey and finally the task is selected with maximum probability of the best performance.

3) The Adaptive Algorithm:

   The Adaptive algorithm is combination of both of these algorithms and it works as per following:
   - During underloaded condition, the algorithm uses EDF algorithm i.e., priority of the job will be decided dynamically depending on its deadline.
   - During overloaded condition, it uses ACO based scheduling algorithm i.e., priority of the jobs will be decided depending on the pheromone value laid on each schedulable task and heuristic function.

   Switching Criteria:

   - Initially the proposed algorithm uses EDF algorithm considering that the condition is not overloaded. But when a job has missed the deadline, it will be identified as overloaded condition and the algorithm will switch to ACO based scheduling algorithm. After 10 jobs have continuously achieved the deadline, again the algorithm will shift to EDF algorithm considering that overloaded condition has been disappeared.

   During underloaded condition, EDF algorithm is used for reducing execution time and during overloaded condition ACO based scheduling algorithm is used for achieving better performance. By this way, adaptive algorithm has taken advantage of both algorithms and overcome their limitations.

III. Simulation Method

We have implemented EDF, ACO based & the adaptive algorithms and have run simulations to accumulate empirical data. We have considered periodic tasks for taking the results. For taking result at each load value, we have generated 200 task sets each one containing 3 to 9 tasks. The results for 35 different values of load are taken ($0.5 \leq$ load ≤ 5) and tested on more than 35,000 tasks. The simulation programs are executed on PIV Intel machine with 512 MB SDRAM and Linux Red Hat Operating System. Here, for periodic tasks load (L) of the system can be determined using the following equation:

$$L = \sum_{i=1}^{m} \frac{E_i}{Q_i} \tag{1}$$

where,

- m = Number of tasks
- E = Execution time required by the task
- P = Period of the task
- D = Deadline of the task
- Q = P if $P \geq D$
  Q = D if $P < D$

The system is said to be overloaded when even a clairvoyant scheduler cannot feasibly schedule the jobs offered to the scheduler [8]. A reasonable way to measure the performance of a scheduling algorithm during an overload is,

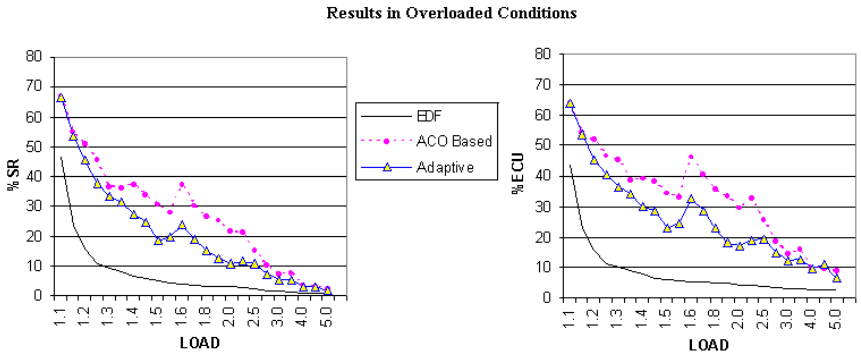

Fig. 1. Load Vs %SR and Load Vs %ECU when load > 1

by the amount of work the scheduler can feasibly schedule according to the algorithm. We have considered following three as our main performance measuring criteria:

1) In real-time systems, deadline meeting is the most important. Therefore, the most appropriate performance metric is the Success Ratio and defined as [10],

$$\text{SR} = \frac{\text{Number of jobs successfully scheduled}}{\text{Total number of jobs arrived}}$$

2) Effective CPU Utilization (ECU) gives information about how efficiently the processor is used and it is defined as,

$$\text{ECU} = \sum_{i \in R} \frac{V_i}{\text{T}}$$

where,

- V is *value of a job* and,
  - *value of a job* = Execution time of a job, if the job completes within its deadline.
  - *value of a job* = 0, if the job fails to meet the deadline.
- R is a set of all the jobs which are executed by the CPU.
- T is total time of scheduling.

3) The execution time required by each scheduling algorithm is very important especially when we are working with real-time systems.

The results are obtained, measured in terms of SR & ECU, compared with EDF algorithm and ACO based scheduling algorithm in the same environment and shown in following section. The execution time taken by each scheduling algorithm is also measured and shown.

IV. Final Results

Table 1 shows the results obtained when Load ≤ 1 using EDF algorithm, ACO based scheduling algorithm and the adaptive algorithm. It proves that the adaptive algorithm is equally optimal for single processor and preemptive environment when system is not overloaded.

Table 1 : RESULTS OBTAINED WITH LOAD ≤ 1

Load	%SR			%ECU		
	EDF	ACO Based	Adaptive	EDF	ACO Based	Adaptive
0.50	100	100	100	49.96	49.97	49.97
0.60	100	100	100	59.88	59.88	59.88
0.70	100	100	100	69.92	69.92	69.92
0.75	100	100	100	74.87	74.87	74.87
0.80	100	100	100	79.87	79.83	79.82
0.85	100	100	100	84.71	84.72	84.71
0.90	100	100	100	89.61	89.61	89.61
0.95	100	100	100	94.54	94.54	94.54
1.00	100	100	100	99.36	99.36	99.37

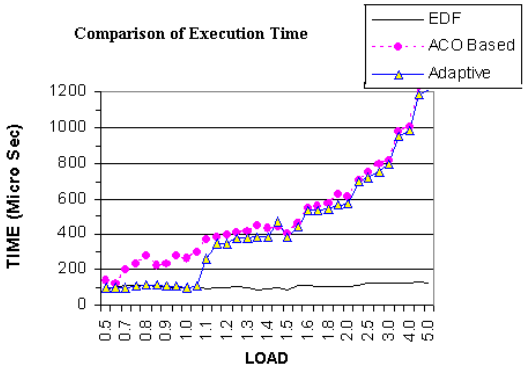

Fig. 2. Load Vs Execution Time (in Micro Seconds)

Fig. 1 shows the results obtained in terms of %SR and %ECU during overloaded conditions, using the same algorithms. Fig. 2 shows the comparison of the execution time taken by each algorithm.

From the results of Fig. 1 and Fig. 2, we can observe that the adaptive algorithm performs better than EDF algorithm during overloaded conditions and it takes less time than ACO based algorithm during underloaded conditions.

V. CONCLUSIONS

The proposed algorithm discussed in this paper is for scheduling periodic tasks on single processor environment when the tasks are preemptive. The results achieved during simulation prove the following:

- The proposed algorithm is equally optimal for single processor, preemptive environment when the system is not overloaded.
- EDF algorithm does not perform well when the system is overloaded and ACO based scheduling algorithm takes more execution time in that type of condition- These are the main limitations of both algorithms. During underloaded condition, the execution time taken by the proposed algorithm is almost same as EDF algorithm (i.e. less time) and during overloaded condition, it gives performance of ACO based scheduling algorithm(i.e. more efficiency).
- The proposed algorithm is dynamic. During simulation only periodic tasks are considered but it can handle aperiodic tasks also. For periodic tasks if arrival time is changed, even though the algorithm works effectively.
- The algorithm can switch automatically between EDF algorithm and ACO based scheduling algorithm. Therefore, the proposed adaptive algorithm is very useful when future workload of the system is unpredictable.

REFERENCES

[1] C.L.Liu and L. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of ACM*, vol. 20(1), pp. 46-61, 1973.

[2] M.Dertouzos and K.Ogata, "Control robotics: The procedural control of physical process", *Proc. IFIP Congress*, 1974.

[3] A.K. Mok, "Fundamental design problems of distributed systems for the hard real-time environment", *Ph.D.Thesis Massachusetts Institute of Technology*, Cambridge, MA, May 1983.

[4] G.Saini, "Application of fuzzy logic to real-time scheduling", *Real Time Conference, 14th IEEE-NPSS*, 2005.

[5] M. Dorigo and G.D.Caro, *New Ideas in Optimization* , McGraw Hill, pp. 11-32, 1999.

[6] V. Ramos, F. Muge and P. Pina,"Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies", *IOS Press*, vol. 87, 2002.

[7] K. Kotecha and A. Shah,"ACO based dynamic scheduling algorithm for real-time operating system", *Sent to AIPR-08, Florida*, 2008.

[8] Jane W.S. Liu, *Real-Time Systems* , Pearson Education, India,pp. 121 & 26, 2001.

[9] M. Dorigo and T.Stutzle, *Ant Colony Optimization* , The MIT Press, Cambridge, 2004.

[10] Krithi Ramamritham, John A Stankovik and Perng Fei Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems", *IEEE Transaction on Parallel and Distributed Systems*, vol. 1(2), April 1990.

[11] Krithi Ramamritham and John A Stankovik, "Scheduling algorithms and real-time support for real-time systems", *Proceedings of the IEEE*, vol. 82(1), January 1994.

[12] C.D. Locke, "Best effort decision making for real-time scheduling", *Ph.D.Thesis, Computer Science Dept., Carnegie-Mellon University*, 1986.

[13] G. Koren and D. Shasha, "D^{over}: An optimal on-line scheduling algorithm for overloaded real-time systems," *SIAM J of Computing*, vol. 24(2), pp. 318-339, April 1995.