2010 International Conference on Computational Intelligence and Communication Networks

Scheduling Algorithm for Real-Time Operating Systems using ACO

Apurva Shah
G H Patel College of Engg & Tech
Charutar Vidya Mandal
Vallabh Vidyanagar, INDIA
e-mail: shahapoorva@ymail.com

Ketan Kotecha
Institute of Technology
Nirma University
Ahmedabad, INDIA
e-mail: drketankotecha@gmail.com

Abstract— **The Ant Colony Optimization algorithms (ACO) are computational models inspired by the collective foraging behavior of ants. By looking at the strengths of ACO, they are the most appropriate for scheduling of tasks in soft real-time systems. In this paper, ACO based scheduling algorithm for real-time operating systems (RTOS) has been proposed.**

During simulation, results are obtained with periodic tasks, measured in terms of Success Ratio & Effective CPU Utilization and compared with Earliest Deadline First (EDF) algorithm in the same environment. It has been observed that the proposed algorithm is equally optimal during underloaded conditions and it performs better during overloaded conditions.

Keywords- ***Real-Time Systems; Scheduling; ACO; 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 correctness but also on the time it takes to produce the result [1]. 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 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 all priorities at design time, and it remains constant for the lifetime of a task. Dynamic algorithms assign priority at runtime, based on execution parameters of tasks. Dynamic scheduling can be either with static priority or dynamic priority. RM (Rate Monotonic) and DM (Deadline Monotonic) are examples of dynamic scheduling with static priority [2]. EDF (Earliest Deadline First) and LST (Least Slack Time First) are examples of dynamic scheduling with dynamic priority. EDF and LST algorithms are optimal under the condition that the jobs are preemptable, there is only one processor and the processor is not overloaded [3][4]. But the limitation of these algorithms is, their performance decreases exponentially if system becomes slightly overloaded [5].

The scheduling is considered as on-line, if scheduler makes scheduling decision without knowledge about the task that will be released in the future. It is proved that, no online scheduling algorithm can achieve competitive factor greater than 0.25 when the system is overloaded. It has been further proved by researchers that the competitive factor of an online scheduling algorithm is at most equal to 0.385 for any system whose loading factor is slightly over one ([6],[7]).

Several characteristics make ACO a unique approach: it is constructive, population-based metaheuristic which exploits an indirect form of memory of previous performance. [8][9]. Therefore in this paper, the same approach has been applied for real-time operating systems.

The whole paper is organized as follows: The system and task model is discussed in Section II. In Section III, the proposed algorithm is explained and discussed. Section IV contains simulation method and performance measuring parameters. Section V contains the results obtained and the paper ends with a brief conclusion in Section VI.

II. SYSTEM AND TASK MODEL

The system knows about the deadline and required computation time of the task when the task is released. The task set is assumed to be preemptive. We have assumed that the system is not having resource contention problem. Moreover, preemption and the scheduling algorithm incur 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 [10]. In a special case of soft real-time systems,

978-0-7695-4254-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CICN.2010.122

called a firm real-time system, there is no value for a task that has missed its deadline, but there is no catastrophe either [11]. Here, we propose an algorithm that applies to firm real-time system. The value of the task has been taken same as its computation time required [12].

III. PROPOSED ALGORITHM

The scheduling algorithm is required to execute when a new task arrives or presently running task completes. The main steps of the proposed algorithm are given as following and the flowchart of the algorithm has been shown in Figure 1:

A. Construct tour of different ants and produce the task execution sequence
B. Analyze the task execution sequences generated for available number of processors
C. Update the value of pheromone
D. Decide probability of each task and select the task for execution

}

A. Tour Construction

First, find probability of each node using equation (2.1). Each schedulable task is considered as a node and probability of each node to be selected for execution is decided using pheromone τ and heuristic value η.

$$p_i(t) = \frac{(\tau_i(t))^{\alpha} * (\eta_i(t))^{\beta}}{\sum_{l \in R_1} (\tau_l(t))^{\alpha} * (\eta_l(t))^{\beta}} \qquad (2.1)$$

where,

- $p_i(t)$ is the probability of ith node at time t; $i \in N_1$ and N_1 is set of nodes (schedulable tasks) at time t.
- $\tau_i(t)$ is pheromone on ith node at time t.
- η_i is heuristic value of ith node at time t, which can be determined by, $\eta_i = \frac{K}{D_i - t}$. (2.2)

  Here, t is current time, K is constant (suitable range is 5 to 10) and Di is absolute deadline of ith node.
- α and β are constants which decide importance of τ and η.

Ants construct their tour based on the value of p of each node as per following:

Ant1: Highest p → second highest p → third highest p →...
Ant2: Second highest p → highest p → third highest p →...
Ant3: Third highest p → highest p → second highest p →...
....

Suppose at time t, there are 4 schedulable tasks. As shown in Figure 1, each task will be considered as a node and from each node; one ant will start its journey. If we consider the priorities of all the nodes are in decreasing order of A,B,C,D; ants will traverse different nodes as per following:

Ant 1: A → B → C → D
Ant 2: B → A → C → D
Ant 3: C → A → B → D
Ant 4: D → A → B → C

B. Analyze the Journey

After all ants have completed their tour, evaluate the performance of different ants' journey. We have analyzed this based on ratio of number of success tasks and number of missed tasks. Find out maximum two best journeys of ants and update the value of pheromone accordingly.

C. Pheromone Update

Pheromone updating on each node is done in two parts:

1. Pheromone Evaporation: Pheromone evaporation is required to forget bad journey of ants and to encourage new paths. Value of τ is updated using
   $$\tau_i = (1-\rho)\tau_i \qquad (2.3)$$
   where,
   - ρ is a constant. (suitable range is 0.2 to 0.4)
   - $i \in R_1$; R_1 is set of all (schedulable and non-schedulable at that time) tasks.
2. Pheromone Laying: Pheromone will be laid only for two best journeys of ants. Select the best journey and put pheromone depending on their order of visited node. Amount of pheromone ($\Delta\tau$) laid will be different at each node i.e. the nearest node will get highest amount of pheromone and far most node will get least.
   $$\tau_i = \tau_i + \Delta\tau_i \qquad (2.4)$$
   where,
   - $i \in N_2$, N_2 is set of nodes traveled by the ant.
   - $\Delta\tau = \frac{ph}{s}$ (2.5)

   Here,
   - $ph = C * \frac{Number\ of\ Sucessed\ Tasks}{Number\ of\ Missed\ Tasks + 1}$ (2.6)
   - s is sequence number of the node visited by the ant during the best journey.
   - Value of C is constant (preferably 0.1).

D. Selection of Task for Execution

After updating pheromone, again find out probability of each node using eq. (2.1) and select the task for execution having the highest probability value

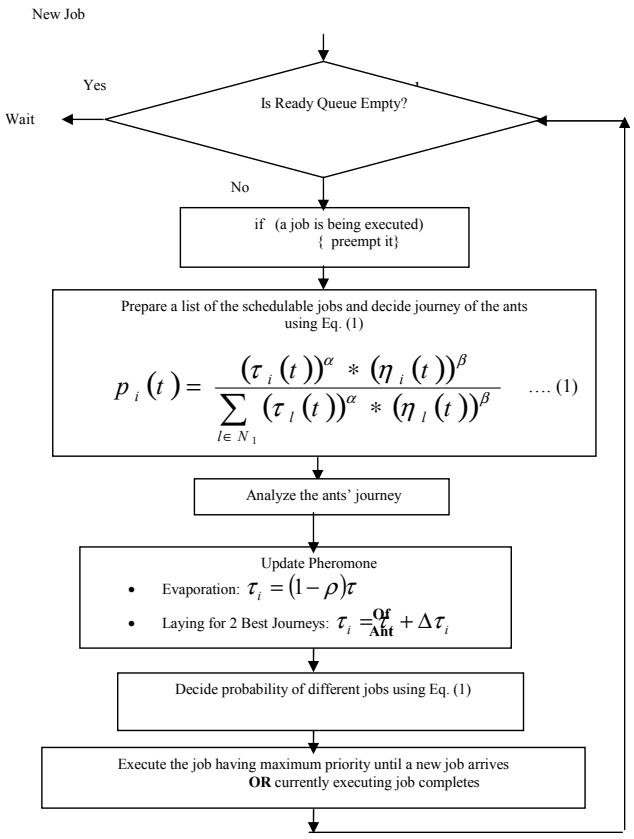

Figure 1. Flowchart of the proposed algorithm

E. Important Points about the Algorithm

- Each schedulable task is considered as a node, and it stores the value of τ i.e. pheromone. Initial value of τ is taken as one for all nodes.
- Value of α and β decide importance of τ and η. During simulation, both values are taken as one
- Number of ants which construct the tour, is important design criteria. During simulation, number of ants taken is same as number of executable tasks the system is having at that time.

IV. A CASE STUDY OF A SCHEDULING INSTANCE

In this section, a scheduling instance consisting of different aperiodic task sets is given and their scheduling with EDF algorithm and the proposed algorithm are

The task set is made up of aperiodic tasks and shown

TABLE I. TASK SET FOR THE CASE STUDY

Task	Arrival Time	Absolute Deadline	Required Exe. Time
A	0	10	3
B	0	10	4
C	0	9	8

Scheduling using EDF Algorithm:

The sequence of scheduling by EDF will be: C, A, B. Therefore only one task achieves deadline.

Scheduling using the proposed Algorithm:

At T = 0, the proposed algorithm works as shown in Table II and the sequence of scheduling will be: A, B, C. Therefore two tasks achieve the deadline and only one misses the deadline.

TABLE II. STEPS OF THE PROPOSED ALGORITHM AT T = 0

i **Tour**	**Path**	**Analyze Journey**		**Imp. After Pheromone Update**
		Succ. Tasks	***Miss. Tasks***	
1	A,B,C	2	1	A(max)
2	B,A,C	2	1	B
3	C,A,B	1	2	C(min)

V. SIMULATION METHOD

We have implemented our algorithm & EDF algorithm and have run simulations to accumulate empirical data. We have considered periodic tasks for taking the results. For periodic tasks, load of the system can be defined as summation of ratio of executable time and period of each task. 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 $\leq$ 5) and tested on more than 35,000 tasks. Results are shown in Table 1 and Figure 2.

The system is said to be overloaded when even a clairvoyant scheduler cannot feasibly schedule the tasks offered to the scheduler. A reasonable way to measure the performance of a scheduling algorithm during an overload is by the amount of work the scheduler can feasibly schedule according to the algorithm. The larger this amount the better the algorithm. Because of this, we have considered following two as our main performance criteria:

1. In real-time systems, deadline meeting is most important and we are interested in finding whether the task is meeting the deadline. Therefore the most appropriate performance metric is the Success Ratio and defined as [13],

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

2. It is important that how efficiently the processors are utilized by the scheduler especially during overloaded conditions. Therefore, the other performance metric is Effective CPU Utilization (ECU) and defined as:,

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

where,

- V is *value of a task* and,
  - *value of a task* = Computation time of a task, if the task completes within its deadline.
  - *value of a task* = 0, if the task fails to meet the deadline.
- R is set of tasks, which are scheduled successfully i.e. completed within their deadline.
- T is total time of scheduling.

An on-line scheduler has a competitive factor C_f if and only if the value of the schedule of any finite sequence of tasks produced by the algorithm is at least C_f times the value of the schedule of the tasks produced by an optimal clairvoyant algorithm [7]. Since maximum value obtained by a clairvoyant scheduling algorithm is a hard problem, we have instead used a rather simplistic upper bound on this maximum value, which is obtained by summing up the value of all tasks [14]. Therefore, we have considered value of ECU for clairvoyant scheduler is 100%.

Finally, the results are obtained, compared with EDF algorithm in the same environment and shown in Table III and Figure 2.

VI. RESULTS

Table III shows the results achieved by the proposed algorithm and EDF algorithm during underloaded conditions. We can observe that the proposed Algorithm is equally optimal for single processor and preemptable environment when system is not overloaded.

TABLE III. RESULTS OBTAINED WITH LOAD ≤ 1

Load	%ECU		%SR	
	EDF	*New Algo.*	*EDF*	*New Algo.*
0.50	49.96	49.97	100	100
0.55	55.04	55.04	100	100
0.60	59.88	59.88	100	100
0.65	64.99	64.99	100	100
0.70	69.92	69.92	100	100
0.75	74.87	74.87	100	100
0.80	79.87	79.87	100	100
0.85	84.71	84.72	100	100
0.90	89.61	89.61	100	100
0.95	94.54	94.54	100	100
1.00	99.36	99.36	100	100

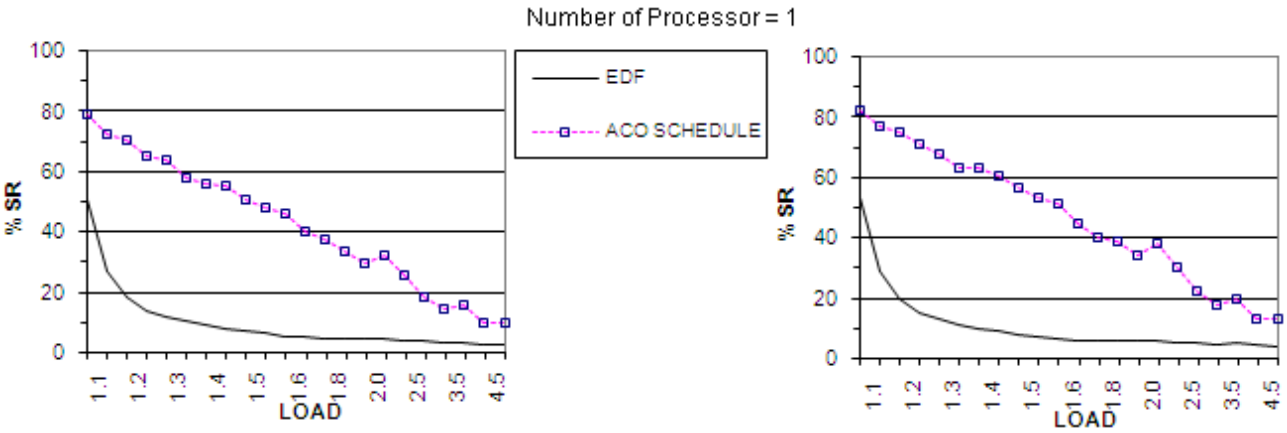

Figure 2. Load Vs. %SR and Load Vs. %ECU when Load > 1

Figure 2 shows the results during overloaded conditions. As expected %SR and %ECU of EDF fall down rapidly but the proposed algorithm works better.

From the values of %ECU and considering maximum value for clairvoyant scheduler, we find that competitive factor of the proposed algorithm is definitely more than 0.595 and 0.425 when load values are 1.25 and 1.50.

Moreover, competitive factor of the proposed scheduling algorithm has been found as 1.00 up to load ≤ 1 i.e. during underloaded conditions.

VII. CONCLUSIONS

The algorithm discussed in this paper is for scheduling of soft real-time system with single processor and preemptive task sets. For scheduling, the concept of ACO has been introduced. The algorithm is simulated with periodic task sets; results are obtained and compared with EDF.

From the results of simulation we can conclude that the proposed algorithm performs equally optimal for single processor, preemptive environment when the system is underloaded. We can also observe that during overloaded conditions performance of EDF is poor, but the proposed algorithm works much better.

REFERENCES

[1] K. Ramamritham and J. A. Stankovik, "Scheduling algorithms and operating support for real-time systems", *Proceedings of the IEEE*, vol. 82, January 1994.

[2] C.L.Liu and L. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment", *Journal of ACM*, January 1973, 20(10): 46-61.

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

[4] A. Mok, "Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment*," Ph.d.thesis*, MIT, Cambridge, Massachusetts, May 1983.

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

[6] S. Baruah, G. Koren, B. Mishra, et al. "On the competitiveness of on-line real-time task scheduling", *In Proceedings of IEEE Real-Time Systems Symposium*, December 1991, pp. 106-115.

[7] J.W.S.Liu, "Real-Time Systems*", Pearson Education*, India, 2001.

[8] M. Dorigo and G. Caro, "The Ant Colony Optimization Metaheuristic in D.Corne, M. Dorigo and F.Glover(eds)", *New Ideas in Optimization,* McGraw Hill, 1999.

[9] 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", *In Second International Conference on Hybrid Intelligent System*, IOS Press, Santiago, 2002.

[10] C. D. Locke, "Best Effort Decision Making for Real-Time Scheduling*", Ph.d.thesis*, Computer Science Department, Carnegie-Mellon University, 1986.

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

[12] A Shah, K Kotecha and D Shah, "Adaptive scheduling algorithm for real-time distributed systems", *To appear in International Journal of Intelligent Computing and Cybernetics* .

[13] K.Ramamritham, J.A.Stankovik, and P.F.Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems", *IEEE Transaction on Parallel and Distributed Systems*, vol. 1, April 1990.

[14] S. Baruah, G. Koren, B. Mishra, A. Raghunath, L. Roiser, and D. Shasha, "On-line scheduling in the presence of overload," *In FOCS*, 1991, pp. 100–110.