International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-8 Issue-12, October 2019

ACO Based Scheduling Method for Soft RTOS with Simulation and Mathematical Proofs

Jay Teraiya, Apurva Shah, Ketan Kotecha

Abstract*: The Ant Colony Optimization (ACO) algorithm is a mathematical model enlivened by the system searching conduct of ants. By taking a gander at the qualities of ACO, it is most suitable for scheduling of tasks in soft real-time systems. In this paper, the ACO based scheduling method for the soft real-time operating system (RTOS) has been profound with mathematical and practical proof. In Mathematical proof, three different Propositions and two Theorems have been given, which prove the correctness of the proposed algorithm. Practical experiments also support mathematical proofs. During the investigation, observations are gathered with different periodic task set. Algorithms have been observed regarding Success Ratio (SR) and Effective CPU utilization (ECU). ACO based scheduling algorithm has been compared with the Earliest Deadline First (EDF) algorithm with parameter SR and ECU. The EDF is dynamic scheduling algorithm and it is most suitable in RTOS when task set is preemptable. It is noted that the new algorithm is equally efficient during under loaded conditions when CPU load is less than one. ACO based scheduling algorithm performs superior during the overloaded conditions when CPU load is more than one where as EDF algorithm performance degraded in overload condition. Empirical study has been executed with a hefty Dataset consist of more than 7500 task set, and a set contains different one to nine processes where CPU load is dynamic for each process set and differ from 0.5 to 5. Algorithms have been executed on five-hundred-time unit for each process set to authenticate the accuracy of both algorithms.

***Keywords:* ACO, EDF, ACO, Real-Time Systems, RTOS**

I. INTRODUCTION

Real-time system is the systems in which the accuracy of the system not only defined by the logical accuracy but also with the time it takes to produce the result. Real-Time systems have decisive, unchanging time restrictions, i.e., the task must be ended within the specified duration; otherwise, the system fails. One can find the existence of two types of real-time systems: Hard and Soft Real-Time System. Hard Real-Time System needs that task deadlines must be met; otherwise, the disastrous situation will arise whereas in Soft Real-Time System, lost an occasional deadline is unwanted but reasonable. Real-time task manager aims to make sure that it meets the deadline for scheduled tasks in the system when we consider the soft real-time system. Vast re-searches are going on real-time task scheduling in order get this desired target. In general, all the real-time systems that exist use preemption and multitasking. Real-Time scheduling methods are widely separated into two methods: Static and Dynamic Methods. Static methods allocate all priorities at design time, and it remains steady for the lifespan of a task. Dynamic methods keep changing the priority at the scheduled time, based on design parameters of any job. Dynamic methods can be endured with static or dynamic priority. Rate Monotonic (RM) and Deadline Monotonic (DM) are the examples of the dynamic scheduling method with static priority [1][2]. There are examples of dynamic scheduling with dynamic priority such as- Earliest Deadline First (EDF) and Least Slack Time First (LST). These algorithms are most the favorable where jobs are preemptable, consist of a single processor, which in turn is under-loaded [3],[4]. However, the constraint of such algorithm is its performance, which diminishes exponentially if the system becomes somewhat overloaded [5].The scheduling is treated as online if the scheduler forges scheduling outcome and doesn't know about the task that is to be released in the future. It is stated that, in an overloaded situation, no other online scheduling algorithm can attain a competitive factor prominent than 0.25. Certainly, many researchers have identified that for any system whose loading factor is nearly equal to 1, the competitive factor of an online scheduling algorithm is nearly equivalent to 0.385 [6],[7]. Certain features make ACO based algorithm an exclusive method: it is effective, population-based metaheuristic that feeds an indirect form of memory of an earlier performance [8][9]. That is one reason why we have considered the same approach for RTOS scheduling.This paper has aimed to formulate as follows: In Section II, the projected algorithm is described and explained. Section III contains mathematical proofs for this algorithm, which includes three Propositions and two Theorems. Section IV illustrates the Simulation method, System and Task Model. Section V represents Results and Discussion and the paper ends with a concise decision in Section VI.

Revised Manuscript Received on October 30, 2019.
***** Correspondence Author
Jay Teraiya*, Computer Engineering Department, Marwadi University, Rajkot, India.
Apurva Shah, Computer Science and Engineering Department, The Maharaja Sayajirao University of Baroda, Baroda, India.
Ketan Kotecha, The Symbiosis Institute of Technology, Pune, India.

II. THE PROPOUND METHOD

The scheduling method is required to plan when a directly running task completes or any new task gets generated. The main steps of the method are shown in subsequent sections, and the consecutive algorithm has been described.

1. Design a journey of distinct ants to yield the better execution sequence of the task.
2. Evaluate the sequences of the task for the given processor.
3. Modify pheromone value.
4. Calculate the probability of all tasks and chose the best task for execution.

A. Creation of Tour

One is required to find the probability of each task using equation in the initial phase. (1) In addition to that, all schedulable tasks are considered as a node and using pheromone τ, and heuristic value η, the probability of all nodes are selected for execution,

$$P_i(t) = \frac{(\tau_i(t))^{\alpha} \times (\eta_i(t))^{\beta}}{\sum_{l \in R_1} (\tau_i(t))^{\alpha} \times (\eta_i(t))^{\beta}} \quad (1)$$

Where,

$P_i(t)$ is the probability of i[th] fork at time t; where i ∈ N_1 and N_1 is a set of the node (schedulable tasks) at time t.

- $\tau_i(t)$ is the value of pheromone of i[th] node at time t.
- η_i is the value of heuristic of i[th] node at time t, which can be regulated as,

$$\eta_i = \frac{K}{D_i - t} \quad (2)$$

Here, t is the current time, K is constant (scale 5 - 10) and D_i is the absolute deadline of i[th] fork.

- α and β are the constants that decide the significance of τ and η.

Ants form their journey based on value p for each fork, as per the following,

- Ant-1: 1[st] maximum p(t) → 2[nd] maximum p(t) → 3[rd] maximum p(t) →
- Ant-2: 2[nd] maximum p(t) → 1[st] maximum p(t) → 3[rd] maximum p(t) →
- Ant-3: 3rd maximum p(t) → 1[st] maximum p(t) → 2[nd] maximum p(t) →

Consider on-time t; there are four tasks schedulable shown in Algorithm 1. Each task will be served as a fork, and from another fork, an ant will start its tour. Let's assume the preference of all the forks is in descending order such as T1, T2, T3, T4; ants will pass over different forks as per the following paths.

- Ant-1: T1→ T2→ T3→ T4
- Ant-2: T2→ T1→ T3→ T4
- Ant-3: T3→ T1→ T2→ T4
- Ant-4: T4→ T1→ T2→ T3

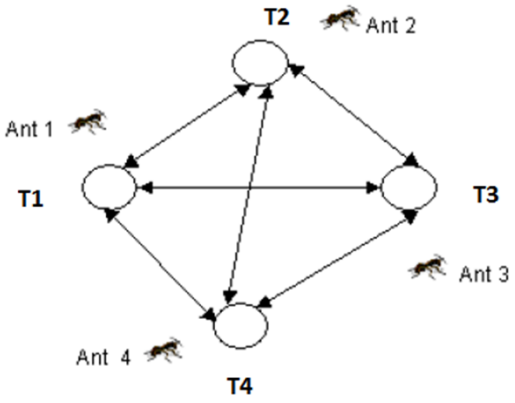

Fig. 1.Ants Journey.

B. ACO Based Algorithm

Once ants have finished their respective journeys, calculate the progress of all ant's journey is calculated. We studied this foundation based on relative number of successful tasks and missed tasks. After that, consider the two leading trips of ants and modify the pheromone cost consequently.

Algorithm 1: ACO Based Scheduling

Input: A set of Processes, Pheromone (τ), Heuristic Value (η), (α, β, ρ) are constants.

Output: Executes the Most Important Process.

for each New Process Arrives or Currently running process complete **do**

if Is Ready Queue is Empty then,
Wait;

/* this step identifies the most suitable process for execution */
Compute **Most_important_Process()** ;

Analyze the Ant's Journey using two tasks:
Success Task = {*Successfully Scheduled: Total Task Arrived*};
Missed Task = {*Unsuccessfully Scheduled: Total Task Arrived*};

/* Update of Pheromone is needed to forget wicked journey of ants */
Compute **Pheromone_update()** to satisfy the Most_Important_Process()

Determine the Probability of each process using Most_Important_Process and execute the process having the highest probability.

end

Most_Important_Process ($P_i(t)$) (*Set of Process P***)** for the i[th] node at time t.

/* Probebility of each task will be calculated based on following equation 1. */

Calculate $P_i(t) = \frac{(\tau_i(t))^{\alpha} \times (\eta_i(t))^{\beta}}{\sum_{l \in R_1} (\tau_i(t))^{\alpha} \times (\eta_i(t))^{\beta}}$

Pheromone_Update (τ)

Calculate Evaporation **(τ_i)** = (1- ρ) τ_i to ignore the lousy path and support new paths.
Calculate Best of two paths to get the Best Path
$(\tau_i) = \tau_i + \Delta\tau_i$

C. Update Pheromone Value

Pheromone update on every node will be done via two different operations:

1. Evaporation Value of Pheromone: Pheromone evaporation is needed to for-get the lousy journey of ants and to support new paths. Value of τ is updated using,

$$\tau_i = (1-\rho)\tau_i \qquad (3)$$

Here,

- ρ is constant (suitable value is 0.2 to 0.4).
- i∈ N_1; N_1 is set of all (schedulable and non-schedulable) task.

2. Value of Pheromone Laying: Pheromone will be adjoined only for two ultimate journeys of ants. Select the most favorable journey and add pheromone to it, based on their order of travelling node. The quantity of pheromone (Δτ) added will be different and vary from node to node, i.e., the possible nearby node will get the highest quantity of pheromone, and the farthest node will get the smallest quantity.

$$\tau_i = \tau_i + \Delta\tau_i \qquad (4)$$

Where,

- i∈ N_2, N_2 is set of nodes travel by the ants.
- $\Delta\tau = \frac{ph}{s}$ (5)

  Here,
  - $ph = C * \frac{Number\ of\ success\ tasks}{Number\ of\ Missed\ tasks+1}$ (6)
  - S is the sequence number of any fork that is hit by the ant during its leading journey.
  - C is a constant (near to 0.1).

D. Selection of Execution Task

After modifying the pheromone value, one needs to compute the possibility of every node bye Eq. 1, then chose the new task for further enactment that has the outrageous value of probability.

E. Algorithm Key Points

- All schedulable tasks are considered as a node, they store τ values, and it is pheromone. The pheromone τ is initialized with value 1 for each node.
- α and β values are decided for the weightage of τ and η. In the experiment, both constants have given equivale weightage which is 1.
- A number of ants which construct the tour is essential in design criteria. During the test, the system is having the same time, and the number of ants decided based on the number of executable tasks.

III. MATHEMATICAL PROOF FOR THE PROJECTED ALGORITHM

The probability of each node will be calculated based on Eq. 1. It will decide which task one should execute to get an optimal result in the proposed algorithm. Following mathematical propositions and theorems have been given with its proof.

Proposition 1: After analyzing journey pheromone will be increased at the rate of $\Delta\tau_i$ (Eq. 4), where $\Delta\tau_i > \Delta\tau_{i+1}$, i∈$N_2$, N_2 is a set of nodes travel by the ants.

Proof - Possible amount of pheromone added to any node after analyzing the journey is $\Delta\tau_i$, Where $\Delta\tau = \frac{ph}{s}$ (Eq. 5) , s is the sequence number of nodes visited by ant during the tour and ph value will be identified based on Eq. 6. Clearly, at first node maximum, possible pheromone is $\frac{ph}{1}$, for the second node it is $\frac{ph}{2}$ and so on. It means the nearest node will get the highest amount of pheromone and far most will get least.

Proposition 2: Pheromone will be decreased at the rate of $(1-\rho)\tau_i$ (Eq. 3) $\forall_i \in N_1$, where ρ is constant and N_1 is the set of schedule and non-schedule task at that time.

Proof - Pheromone evaporation is required to forget the lousy journey of ant and to encourage new paths. Possible amount of pheromone decreases to any node after analyzing the journey is$(1-\rho)\tau_i$.

Theorem 1: Let P be the probability that the algorithm finds an optimal solution within the first analyzing journey, then for an arbitrary small $\epsilon > 0$, $P \geq 1-\epsilon$. By definition$P_{max} = 1$.

Proof - For best two journeys, $i \in N_1 \cap N_2$, where i is the task which is part of both ant journey then pheromone lying will be done on i is $\Delta\tau_i$ as per proposition-1 and according to Eq. 1, the probability P_i will increase.

If $i \notin N_2$ and $i \in N_1$ then pheromone value τ_i will continuously decreasing and it will help us to ***forget*** a bad journey. Due to pheromone trail limits τ_{min} and τ_{max} ***one*** can guarantee that any feasible choice in Eq. 1, for any solution is made with a probability$P_{min} > 0$ [15]. At trivial lower bound for

$$P_{min} \geq \frac{\tau_{min}^{\alpha}}{(N_1-1)\tau_{max}^{\alpha} + \tau_{min}^{\alpha}} \qquad (7)$$

Proposition 3: Once an optimal solution has been found for any task such that $\notin N_1$, it holds that $\tau_i = 0$.

Proof - After the execution of the task, the task will not belong to the optimal solution and do not receive pheromone anymore.

Theorem 2: The probabilistic decision taken by ant will be biased when incorporating heuristic information into an ACO based solution.

Proof - Prior available information on the schedulable task can be used to derive heuristic information that biases the probabilistic decision taken by the ant (Eq.2). When assimilating such heuristic information into ACO solution, the favorable choice is $F_i(\tau_i) = (\tau_i)^{\alpha} \times (\eta_i)^{\beta}$. Based on Eq. 1 and Eq. 2 η_i measures the heuristic desirability of choosing a solution as a task i. Infect,

Theorem-1 are not going to affected by the heuristic information, η is limited to some (instant specific) interval $[\eta_{min}, \eta_{max}]$ with $\eta_{min} > 0$ and $\eta_{max} < +\infty$. Then the heuristic information only affected of changing the lower bound of the probability P_{min} of making a specific decision.

IV. SIMULATION METHOD, SYSTEM AND TASK MODEL

When the task is released, we pretend that the system already knows about the task deadline and imperative data to figure out the required time to complete the task. The task set is considered to be preemptive and pretending that the system doesn't have any constraint of resource clash. Furthermore, it also considered that scheduling and preemption do not have any other overhead. In Soft RTOS, each task possesses a positive value. The simple ideology is to yield as much benefit as possible. If a particular task succeeds, then the system contemplates its benefits; otherwise, the system attains less benefit from the task [10]. In a distinct case of firm real-time system, that suggest if any task missed its deadline, then no value will be mediated, but there is no collapse as well [11]. With this work, we propound an algorithm which affixes to the firm real-time system, and the value of the task has been treated very similar to that of its required computation time.[12].

This paper analyses proposed algorithm with the EDF algorithm and execute the simulations to gather the experimental outcomes; also, we considered periodic tasks in order to get effective results. For that, a system load can be described as the aggregate of the ratio of executable time and the time of each task. In order to achieve effective results, at every load value we have produced 7500 task sets and every load contains utmost 1 to 9 tasks. The outcomes from this experiment contain different values of load (ranges from 0.5 - 5)**,** and it examined on 35,000+ task. Moreover**,** the results of this phenomena are revealed in Table 3 and Figure 3 [16]. Higher the amount of work ***is*** scheduled, the better and competent the algorithm is. For this reason, we have measured the two of our main performance metrics:

1. In RTOS, meeting the deadline is utmost significant and crucial, and therefore, we are more concerned towards result, whether the task is meeting the deadline or not. Based on that, the most reliable metric that we get is Success Ratio (SR), and is defined as [13],

$$SR = \frac{Number\ of\ Task\ successfully\ scheduled}{Total\ Number\ of\ Task\ arrived} \quad (8)$$

2. It is potentially important to know how effectively the scheduler exploits the processes, peculiarly during heavy load condition. Therefore, we also considered other performance metrics such as Effective CPU utilization (ECU) and is defined in [16],

$$ECU = \sum_{i \in R} \frac{V_i}{T} \quad (9)$$

Where,

- V is the task value and,
  - Task Value = Estimated time of the task if the task accomplishes its work within its deadline.
  - Task Value = 0 if the task fails in order to meet its deadline.
- R is a task set, which is scheduled profitably, i.e., executed within its deadline.
- T is the scheduling total time.

Table- I: Result Obtained with Load <= 1

Load	%ECU		%SR	
	EDF Algorithm	*ACO Based Algorithm*	*EDF Algorithm*	*ACO Based Algorithm*
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

An online scheduler has a competitive factor C_f that exist if and only if the value of the schedule of any finite sequence of tasks formed by the algorithm is at least C_f times the value of the schedule of the tasks formed by an optimal clairvoyant algorithm [7]. Since maximum value, seized by a clairvoyant scheduling algorithm is a hard problem, therefore we have instead used a rather condensed upper bound on this maximum value, which can be obtained by summation of the value of all tasks [14]. Hence, for the clairvoyant scheduler, we have considered the value of ECU as 100%.

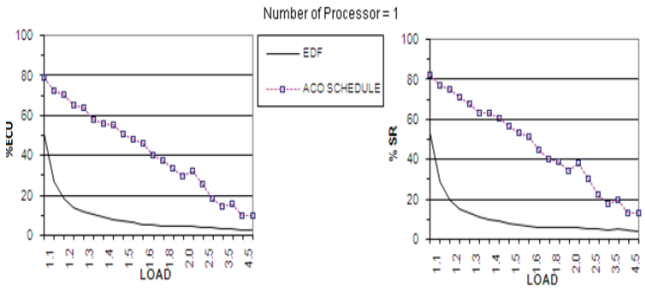

Fig. 2. CPU Load Vs. %ECU and CPU Load Vs.

V. RESULTS AND DISCUSSION

From the empirical study, it is perceived that when the system is not heavily loaded, our projected algorithm gives an ideal result for a unified processor and the preemptive conditions. Table 1 displays the outcomes achieved by our algorithm and the EDF algorithm under loaded conditions. In addition to that, Fig. 2. specifies the results of an overloaded condition.

Furthermore, presumed %SR and %ECU of EDF drop quickly; however, our algorithm works prominently and gives efficient progress. The values of %ECU and the maximum value of the clairvoyant scheduler, we notice that the competitive factor of our algorithm is greater than 0.595 and 0.425 when loads are 1.25 and 1.50. Furthermore, in under loaded conditions, the competitive factor of our scheduling has been found to 1.00 and up to load ≤ 1.

VI. CONCLUSION

In this work, an algorithm specifically for the scheduling of a soft real-time system with a unified processor and the preemptive task have been introduced. In addition to that, for scheduling, ACO has been motivated and introduced. The projected method is implemented with a periodic task, and cumulative outcomes are gathered and collate it with EDF. From the mathematical proof, shown in this work and the results of the experiment, this paper concluded that the projected method accomplishes equally best for a single processor, preemptive conditions when the system is heavily load-ed. This paper has also monitor and analyze the performance of EDF that significantly diminished, during maximum loaded conditions; however, the profound algorithm works in a much better way. So, for real time scheduling it is possible to use swarm techniques for batter performance in underload as well as in overload scenario. In future more Swarm Intelligence methods like PSO, GA etc... can be explored to implement Soft Real Time Schedulers.

ACKNOWLEDGMENT

We would like to acknowledge and give our sincere thanks to leading India ai group which is a nationwide initiative on “AI and deep learning Skilling and Research”. It is funded by Royal Academy of Engineering, UK under Newton Bhabha.

REFERENCES

1. J. Teraiya and A. Shah, “Comparative Study of LST and SJF Scheduling Algorithm in Soft Real-Time System with its Implementation and Analysis”, in International Conference on Advances in Computing, Communications and Informatics (ICACCI), Banglor, India, Proceeding in the IEEE Xplore, pp. 706-711, 2018.
2. C. Liu and J. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”, in Journal of the ACM, vol. 20, no. 1, pp. 46-61, 1973.
3. M.Dertouzos and K.Ogata, “Control robotics: The procedural control of physical process,” in Proceedings IFIP Congress (IFIP'74), pp. 807-813, 1974.
4. A. Mok and A. Ka-Lau, “Fundamental design problems of distributed systems for the hard-real-time environment”, Thesis (Ph. D.), Massachusetts Institute of Technology, Cambridge, 1983.
5. G. Saini, “Application of fuzzy logic to real-time scheduling”, in 14th IEEE-NPSS Real Time Conference, Stockholm, Sweden, pp. 113-116, 2005.
6. S. Baruah, G. Koren and B. Mishra, “On the competitiveness of on-line real-time task scheduling”, in IEEE Proceedings 12th Real-Time Systems Symposium, San Antonio, TX, USA, pp. 106-115, 1991.
7. J. Liu, Real-time systems. Upper Saddle River, N.J.: Prentice Hall, 2009.
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, pp-13-49, 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,” Soft Computing Systems – Design, Management and Applications, inProceedings of the 2nd International Conference on Hybrid Intelligent System, IOS Press, Santiago, 2002.
10. Carey, Douglass, Locke “Best-effort decision-making for real-time scheduling”, Doctoral Dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1986.
11. G.Koren and D.Shasha, “Dover: An optimal on-line scheduling algorithm for overloaded real-time systems,” in SIAM Journal of Computing, Vol. 24(2), pp. 318-339, April 1995.
12. A Shah , “Adaptive scheduling algorithm for real-time distributed systems ”, in Biologically-Inspired Techniques for Knowledge Discovery and Data Mining pp. 236-248, 2014.
13. K. Ramamritham, J. Stankovic and P. Shiah, “Efficient scheduling algorithms for real-time multiprocessor systems”, in IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 2, pp. 184-194, 1990.
14. S. Baruah, G. Koren, B. Mishra, A. Raghunath, L. Roiser and D. Shasha, “On-line scheduling in the presence of overload”, in Proceedings 32nd Annual Symposium of Foundations of Computer Science, San Juan, Puerto Rico, USA, pp. 100-110, 1991.
15. M. Dorigo and T. Stützle, “Ant colony optimization”. Cambridge, Mass.: MIT Press, pp. 131-132, 2004.
16. K. Kotecha and A. Shah, “Scheduling Algorithm for Real-Time Operating Systems Using ACO.”, in International Conference on Computational Intelligence and Communication Networks, Bhopal, India, pp.617-621, 2010.

AUTHORS PROFILE

Jay Teraiya has completed Bachelor of Engineering from GCET – Vallabh Vidyanagar under S. P. University. He has also completed his M. S. in Software Engineering from BITS Pillani. He is pursuing in Ph. D from the M. S. University of Baroda under the guidance of Dr. Apurva Shah.

Dr. Apurva Shah, Associate Professor and Head of Department (Computer Science and Engineering) in Faculty of Technology, the M. S. University of Baroda Gujarat. He is also director of Computer Center in the University. His area of interest are Real Time System, Artificial intelligence and distributed computing. He has completed his Ph. D. from S. P. University Vallabh Vidyanagar.

Dr. Ketan Kotecha, Professor, Computer Science & Engineering, Head, Symbiosis Centre for Applied Artificial Intelligence (SCAAI) Dean, Faculty of Engineering, Symbiosis International (Deemed University) Director, Symbiosis Institute of Technology Chief Executive Officer (CEO), Symbiosis Centre for Entrepreneurship and Innovation TEDx speaker 2015 | Author – Introduction to Critical Thinking (Macmillan) Recipient of Erasmus + faculty mobility grants from European Union.