# 6 THEORETICAL PROOF FOR ACO BASED SCHEDULING ALGORITHM

Real-time task scheduling aims to ensure that it meets the deadline for scheduled tasks in the soft real-time system. Vast re-searches are going on real-time task scheduling to get this desired target. In general, all the real-time systems that exist use pre-emption 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 the design parameters of any job. Rate Monotonic (RM) and Deadline Monotonic (DM) are examples of static priority scheduling algorithms [15][16]. 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 favourable where jobs are preemptable, consist of a single processor, which in turn is under-loaded [54]. However, the constraint of such an algorithm is its performance, which diminishes exponentially if the system becomes somewhat overloaded [18]. The scheduling is treated as online if the scheduling algorithm forges scheduling outcome and doesn't know about the task to be released in the future. Certain features make ACO based algorithm an exclusive method: it is an effective, population-based metaheuristic that feeds an indirect form of memory of an earlier performance[55][56][57]. That is one reason why it has considered the same approach for Real-Time scheduling.

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 tasks in soft real-time systems [33][58][59]. In this thesis, 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.[21]. Based on mathematical proof, it has been again demonstrated the effectiveness of the ACO-based scheduling algorithm [44].

## 6.1 ACO Based Scheduling

The scheduling method must execute 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 a 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 choosing the best task for execution.

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

### 6.1.1 Creation of Tour

One is required to find the probability of each task using equation no 5 in the initial phase. 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,

$$P_i(t) = \frac{(\tau_i(t))^{\alpha} \times (\eta_i(t))^{\beta}}{\sum_{l \in R_1} (\tau_i(t))^{\alpha} \times (\eta_i(t))^{\beta}} \qquad (5)$$

Where,

$P_i(t)$ is the probability of $i^{th}$ fork at time t; where $i \in N_1$, 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.
- $\eta_i$ is the value of heuristic of $i^{th}$ node at time t, which can be regulated as,

$$\eta_i = \frac{K}{D_i - t} \qquad (6)$$

  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 the value $P_i(t)$ 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 schedulable tasks shown in Figure 6.1. 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.

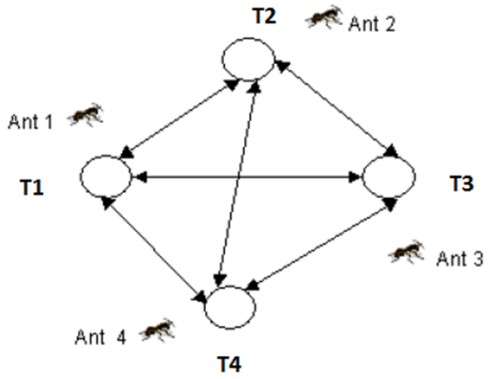


Figure 6.1 – Ant Journey

### 6.1.2 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 forget ants' lousy journey and support new paths. The value of τ is updated using,

$$\tau_i = (1-\rho)\tau_i \qquad (7)$$

Here,

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

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 pheromone (Δτ) added quantity will be different and vary from node to node, i.e., the possible nearby node will get the highest amount of pheromone, and the farthest node will get the smallest quantity.

$$\tau_i = \tau_i + \Delta\tau_i \qquad (8)$$

where,

- i ∈ $N_2$, $N_2$ is a set of nodes travel by the ants.
- $\Delta\tau = \frac{ph}{s}$ (9)

  Here,

  - $ph = C * \frac{Number\ of\ success\ tasks}{Number\ of\ Missed\ tasks+1}$ (10)
  - S is the sequence number of any fork that the ant hits during its leading journey.
  - C is a constant (near to 0.1).

### 6.1.3 Selection of Task

After updating pheromone, find out the probability of each node using equation 5 and select the task for execution having the highest probability value. Thus, the complete flow of the scheduling algorithm has been given in figure 6.2.

### 6.1.4 Algorithm Key Points

1) All schedulable tasks are considered as a node, they store $\tau$ values, and it is pheromone. The pheromone $\tau$ is initialized with a value of 1 for each node.

2) $\alpha$ and $\beta$ values are decided for the weightage of $\tau$ and $\eta$. In the experiment, both constants have given equal weightage, which is 1.

3) The 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 is decided based on the number of executable tasks.

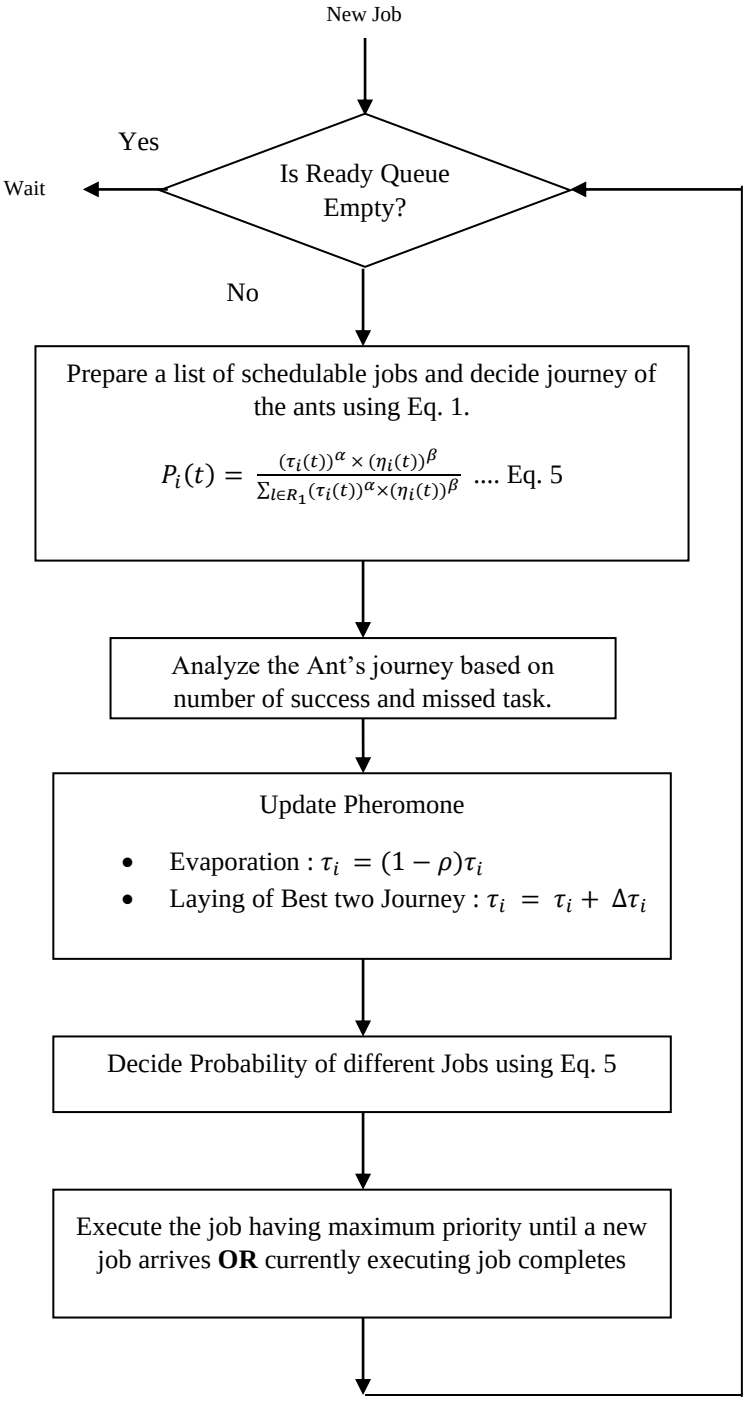


Figure 6.2 – Flow of the ACO Based Scheduling Algorithm

## 6.2 ACO Based Scheduling Algorithm

ACO-based scheduling algorithm invokes when currently running task is completed, or new task arrives in the queue. This algorithm considers all task which is ready to run. It Calculate $P_i(t)$ values for each task using the procedure Most_important_Task(), and the procedure uses equation 5 to calculate the probability. The algorithm analyzes the ant journey based on the number of successful tasks and missed tasks from the total task arrived. The algorithm updates the pheromone values using Pheromone_update(); it is required to forget the lousy journey. Once ants have finished their respective journeys, calculate the progress of all ant's journeys. The study of this foundation is based on a relative number of successful tasks and missed tasks. After that, consider the two leading trips of ants and modify the pheromone cost consequently. After this algorithm, determine the Probability of each task using the Most_Important_Task() procedure and execute the task having the highest probability. This algorithm executes when the selection of task decision needs to take.

---

**ACO Based Scheduling Algorithm:**

---

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

**Output:** Executes the Most Important Task.

**for each** New Task that Arrives or Currently running Task complete, **do**

**if** Is Ready Queue is Empty then,

Wait;

/* this step identifies the most suitable task for execution */

Compute **Most_important_Task()** ;

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 the wicked journey of ants */

Compute **Pheromone_update()** to satisfy the Most_Important_Task()

Determine the Probability of each task using Most_Important_Task and execute the task having the highest probability.

**End for**

---

**Procedure - Most_Important_Task ($P_i(t)$)**

---

Calculate $P_i(t)$ for each task in task Set P at time t.

/* Probability of each task will be calculated based on the following equation */

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

---

**Procedure - Pheromone_Update ($\tau$)**

---

Calculate Evaporation **($\tau_i$) =** (1- $\rho$) $\tau_i$ to ignore the lousy path and support new paths.

Calculate the Best of two paths to get the Best Path $(\boldsymbol{\tau_i}) = \tau_i + \Delta\tau_i$

---

## 6.3 Mathematical Proof for Algorithm

The probability of each node will be calculated based on Eq. 5. Then, it will decide which task should execute to get an optimal result in the proposed algorithm. The following mathematical propositions and theorems have been given with their proof.

**Proposition 1:** After analyzing the journey, pheromone will be increased at the rate of $\Delta\tau_i$ (Eq. 8), where $\Delta\tau_i > \Delta\tau_{i+1}$, i$\in N_2$, $N_2$ 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. 9), s is the sequence number of nodes visited by ant during the tour, and $ph$ value will be identified based on Eq. 10. Clearly, at the first node, the 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 pheromone, and far most will get the least.

**Proposition 2:** Pheromone will be decreased at the rate of $(1-\rho)\tau_i$ (Eq. 7) $\forall_i \in N_1$, where ρ is constant and $N_1$ is the set of schedule and non-schedule tasks at that time.

**Proof -** Pheromone evaporation is required to forget the lousy journey of ants and to encourage new paths. The 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 the 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. 5, the probability $P_i$ will increase.

If $i \notin N_2$ and $i \in N_1$ then pheromone value $\tau_i$ will continuously decreasing and it will help us to ***forget*** a bad journey. Due to pheromone trail limits $\tau_{min}$ and $\tau_{max}$ ***one*** can guarantee that any feasible choice in Eq. 5, for any solution is made with a probability $P_{min} > 0$ [58]. At trivial lower bound for

$$P_{min} \geq \frac{\tau_{min}^{\alpha}}{(N_1 - 1)\tau_{max}^{\alpha} + \tau_{min}^{\alpha}} \qquad (11)$$

**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.6). 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. 5 and Eq. 6 $\eta_i$ measures the heuristic desirability of choosing a solution as a task i. Infect, Theorem-1 are not going to be 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 changing the lower bound of the probability $P_{min}$ of making a specific decision [58].

## 6.4 Conclusion

The empirical study perceives that, projected algorithm gives an ideal performance for a unified processor and the pre-emptive conditions when the system is not heavily loaded. In addition to that, the mathematical proof shows the effectiveness of ACO-based scheduling algorithm in Soft Real-Time systems. So, for real-time scheduling, it is possible to use swarm techniques for batter performance in underload as well as in overload scenarios. In the future, more Swarm Intelligence methods like PSO, GA, etc. can be explored to implement Soft Real-Time scheduling algorithms.