Scheduling Optimization for Soft Real-Time Single Processor Systems using Swarm Intelligence Techniques

A SYNOPSIS

Submitted in partial fulfillment of the

requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE & ENGINEERING

By

JAY BHIKHALAL TERAIYA

Under Guidance of

DR. APURVA SHAH

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

FACULTY OF TECHNOLOGY & ENGINEERING

THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA

VADODARA-390002 (INDIA)

February 2021

ABSTRACT

The Real-Time System is the system in which the system's accuracy is 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. Scheduling a task in Real-Time System is always a challenging problem. Researchers have targeted the Scheduling of Soft Real-Time Single Processor System in this research. The scheduler can be divided into static and dynamic, which depend on the priority they follow. The static algorithm uses a unique priority to each task during the scheduling activity, whereas the dynamic algorithm priority changes during the scheduling process. Researchers have done a critical analysis of many Static and Dynamic scheduler and represented in this research. Based on critical analysis, researchers have also proposed a hybrid scheduler call S_LST. This scheduler overcomes the drawback of the Static and Dynamic scheduler. In the recent past, different researchers have proposed a Swarm Intelligence based scheduling algorithm. Researchers have studied the ACO based scheduling algorithm and given mathematical proof for ACO based scheduling algorithm in this research. Researchers have explored more Swarm Technics and identified that Particle Swarm Optimization could effectively solve Scheduling in Soft Real-Time System. Researchers have designed and developed the PSO based scheduling algorithm for Soft Real-Time Single Processor System. The proposed algorithm has been compared with the existing EDF and ACO based scheduler with respect to the Effective CPU (ECU) and Success Ratio (SR) parameter. It has been found that the proposed PSO based scheduler performs more effective than EDF and ACO based scheduler. To validate the correctness of this algorithm, researchers have considered a large dataset of the periodic task. Research has considered 6800 tasks set, which vary in terms of CPU load and the number of tasks within the task set. CPU load ranges from 0.5 to 5.0, and the number of task set varies from 1 to 9.

TABLE OF CONTENTS

1 INTRODUCTION

This chapter briefly introduces the Real-Time System. It briefs about the Type of Real-Time System, Type of Task, and Type of Scheduling algorithm to schedule the given task set. It also tells about the motive behind this work, the problem statement, and the research contributions.

1.1 The Real-Time System

Real-Time systems have become part of human life to complete their day to day needs. Real-Time System has many applications surrounding us like digital control systems, flight control, vehicle control, healthcare devices, IoT devices, and many more. In the 21st century, usage of Real-Time systems has increased widely. Like a conventional operating system, we also use Real-Time systems in our day to day life, but when real-time systems work well, they make us forget their existence. Real-Time System focuses on completing the task before its deadline, whereas the conventional operating system tries to give minimum response time for any given time. There is always a specific deadline associated with Real-Time Task, whereas a typical task does not have any particular timeframe. Text Editor, Browser, music players are examples of such typical applications, whereas Smart Watch, aircraft control, and missile control systems are examples of Real-Time applications [1].

The Real-Time System is the system in which the system's accuracy is 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. Since the last few years, Real-Time systems usage has been increasing in time-critical applications. Designing systems that are expected to deliver real-time results involves an equal emphasis on managing timing constraints of various functionalities of the system. Processes in the real-time system have defined deadlines and need to complete the process within its deadline. Real-time systems need a scheduling algorithm that assigns the tasks to the processor by considering the deadline constraints and supporting other scheduling requirements.

1.2 Type of Real-Time System

Real-Time System is divided into three categories: Hard Real-Time, Soft Real-Time, and Firm Real-Time System based on their timing constraints. Real-Time systems will be considered Hard Real-Time System if the failure to meet its deadline is a fatal fault. A few examples of Hard Real-Time Systems are Metro Train and its signal system, Missile technology, Flight control system. In contrast, the Soft Real-Time System with few misses of the deadline does not cause serious harm; only the system's overall performance becomes poorer when more jobs miss their deadline. A few examples of Soft Real-Time systems include ATM systems, Mobile applications, and telephone switches. If a task misses its deadline in the Firm Real-Time System, then the result of the given task will be ignored [2].

1.3 Type of Task in Real-Time System

The real-time system has three kinds of task models call Periodic, Aperiodic, and Sporadic tasks. In the periodic task, each task is generated at regular time intervals. The Real-Time System is invariably required to respond to external events and to respond; it executes aperiodic or sporadic tasks whose release times are not known to the system in advance. The interarrival times between consecutive tasks in such a task may vary widely, and, in particular, it can be arbitrarily small. We call the task is aperiodic if the process in it have soft deadlines. Each unit of work is scheduled and executed by the system as a task.

Each task has a different characteristic, like release time, deadline, period, and execution time. The release time of a task is the instant of time at which the task becomes available for execution. The task can be scheduled and executed at any time after its release. The deadline for a task is the instant of time by which its execution needs to be completed. A task's deadline is sometimes called an absolute deadline, which is equal to its release time plus its relative deadline. The execution time of any task is considered the unit amount of time required to execute it on the processor. If the task is periodic, then the task's period indicates the occurrence interval of the given task [3]. The task set can also be divided into preemptive and non-preemptive as well.

1.4 Type of Scheduling Algorithm

In Real-Time Systems, selecting the scheduling algorithm is a critical task, and it will be decided based on the characteristics of the Real-Time System and the task type [4]. Real-Time scheduling methods are widely separated into two categories: Static priority and Dynamic priority scheduling methods. Static scheduling methods allocate all priorities at design time, and it remains steady for the lifespan of a task. Dynamic scheduling methods keep changing the priority at the scheduled time based on any task's design parameters. Dynamic methods can be endured with static or dynamic priority. Rate Monotonic (RM) and Deadline Monotonic (DM) are examples of the dynamic scheduling method with static priority [5][6]. There are examples of dynamic scheduling with dynamic priority, such as the Earliest Deadline First (EDF) and Least Slack Time First (LST). These algorithms are most favorable where jobs are preemptable and consist of a single processor, which is under-loaded [6]. However, such an algorithm's constraint is its performance, which diminishes exponentially if the system becomes somewhat overloaded [7]. The scheduling is treated as online if the scheduler forges scheduling outcomes 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, an online scheduling algorithm's competitive factor is nearly equivalent to 0.385 [8].

1.5 The motivation for This Work

In Real-Time Operating System, selecting the scheduling algorithm is a critical task, and it will be decided based on the characteristics of the system and the process type. The scheduler can be divided into static and dynamic, which depends on the priority they follow in selecting the process for execution. The static algorithm uses a unique priority to each process throughout the scheduling, and the dynamic algorithm priority changes during the scheduling process. Many researchers have proposed different static and dynamic scheduling algorithm. It has been observed that static schedulers perform well during overload scenarios, and dynamic algorithms

perform better than static schedulers in the underload scenario. The Ant Colony Optimization (ACO) based scheduling algorithm was proposed for Soft Real-Time System in 2009.

Researchers have given an entirely new direction for scheduling tasks using Artificial Intelligence and Swarm techniques. Swarm intelligence is the study of computational systems inspired by collective intelligence. Collective Intelligence emerges through the cooperation of large numbers of homogeneous agents in the environment. Examples include schools of fish, flocks of birds, and colonies of ants. Such intelligence is decentralized, self-organizing, and distributed throughout an environment. Using Swarm intelligence, it is possible to find optimal solutions for scheduling the task [9][10]. The researcher has proposed ACO based scheduling algorithm, and it has been shown that the swarm-based scheduling algorithm performs equally well as the Dynamic scheduler. It gives better performance in an overload scenario as well. The ACO based scheduling method for the soft real-time operating system (RTOS) has been profound with practical proof [11].

Many swarm intelligence techniques like the Bees Algorithm, Particle Swarm Optimization, Intelligent water drop, Gravitational Search Algorithm, and many more. They are still not explored with Soft Real-Time System, specifically in scheduling task problem. Researchers have studied all swarm techniques and focused on the Particle Swarm Optimization technique to develop the Soft Real-Time System's new scheduling algorithm.

1.6 Problem Statement, Objectives, and Research Contribution

1.6.1 Problem Statement

To design and develop the scheduling algorithm for Soft Real-Time Single Processor System using Swarm Intelligence Techniques.

1.6.2 Objectives

In order to achieve our goal, we planned:

1) Analyze the existing Static and Dynamic scheduling algorithm and observe their performance in overload and underload scenario.
2) Design of a hybrid scheduling algorithm using the characteristics of Static and Dynamic scheduling algorithm.
3) Give mathematical proof of ACO based scheduling algorithm.
4) Explore different Swarm Intelligence Techniques for Scheduling in Soft Real-Time System.
5) Identify the Swarm Technique and develop a scheduling algorithm for Soft Real-Time Single Processor System.

1.6.3 Research Contribution

For designing and developing of Swarm Based Scheduling algorithm, the following are our research contributions through this work

1) Critical analysis has been done for many Static and Dynamic scheduling algorithms for the Soft Real-Time Single processor system. These analyses have been published in paper publication no. 1, 2, and 3.
2) Based on the analysis of the Static and Dynamic scheduling algorithm, researchers have proposed Hybrid Scheduler (S_LST) for Soft Real-Time System. This work has been published in paper publication no. 4.
3) Researchers have studied the existing ACO based scheduling algorithm design and developed by Dr. Apurva Shah and derived the mathematical proof for the same. This work has been published in paper publication no. 5.
4) Researchers have done a comprehensive study on Swarm Intelligence to identify suitable swarm techniques for scheduling purposes in the Soft Real-Time Single Processor system.
5) Researchers have designed and developed the Optimized Scheduling Algorithm for Soft Real-Time Single Processor System using Particle Swarm Optimization Technique. This work has been published in paper publication no. 6.

2 LITERATURE STUDY AND RELATED WORK

In Soft Real-Time Single Processor System, typically, schedulers are divided into two major categories like static and dynamic schedulers, which depend on the priority they follow. The static algorithm uses a unique priority to each process throw out the scheduling, whereas the dynamic algorithm priority changes during the scheduling process [12][13]. Researchers have also proposed hybrid scheduling algorithms which use the characteristic of the static and dynamic scheduling algorithm. In the recent past, genetic algorithms for scheduling has been proposed by researchers. ACO based scheduling algorithm for Soft Real-Time System has changed the complete direction of solving the scheduling problem. [11].

2.1 The Dynamic Scheduling Algorithms

Dynamic schedulers make decisions during the runtime of the system. This allows not only to design a more flexible system but also to associate calculation overhead with it. The dynamic schedulers decide what task to execute depending on the importance of the task, called priority. The task priority may change during the runtime[14][15]. The dynamic schedulers widely used with the Soft Real-Time single processor system are the earliest deadline first (EDF) and Least Slack Time First (LST). These algorithms are described as follows.

1) EDF - The earliest deadline first is a dynamic scheduling algorithm, which gives the highest priority to the task, which has the nearest absolute deadline. Priorities of tasks are allocated dynamically and inversely proportional to the active processes' absolute deadlines [16]. The algorithm executes when the current process completes or the new process arrives.
2) LST - The LST is a dynamic scheduling algorithm that gives maximum priority to the process, which has the shortest slack time. The slack time (l) can be calculated at time t with the deadline interval d and remaining execution time c [17]. The algorithm executes when the current process completes or the new process arrives.

2.2 The Static Scheduling Algorithms

The static scheduler can calculate the order of execution before runtime as well. The static scheduler also decides the sequence of tasks based on priority, but the priority value will not change during runtime [18]. The static schedulers widely used with the Soft Real-Time Single processor are Rate Monotonic (RM) and Shortest Job First (SJF). These algorithms are described as follows.

1) RM - The rate monotonic is a static scheduling algorithm that gives maximum priority to the process, which has the smallest period or lowest rate [16][19]. The rate of a process is already known in RTOS and is defined as the task occurs again in a given duration. The algorithm executes when the current process completes or the new process arrives.
2) SJF - The shortest job first algorithm is a static scheduling algorithm that gives maximum priority to the process, which has the shortest execution time [19]. The execution time of a process is already known in RTOS and is defined as the process requiring CPU time to complete the given task. The algorithm executes when the current process completes or the new process arrives.

2.3 The Hybrid Scheduling Algorithms

The Dynamic algorithms perform well in the underload scenario and schedule all processes in a given process set. The static algorithms perform well in overload scenarios and try to schedule the maximum process in a given process set. The ideal algorithm can be designed, which uses dynamic and static algorithms, and it performs well in underload and overload scenarios. Based on this concept, many Hybrid Scheduler has been introduced by researchers. A few of them have been explained below.

1) O_EDF and R_EDF - EDF algorithm performs well when the system is not overloaded, but its performance decreases exponentially when it becomes slightly overloaded. The researcher has applied certain modifications in the conventional EDF algorithm to overcome this limitation and proposed two algorithms named O_EDF and R_EDF [20].

2) D_EDF - This scheduling algorithm overcomes the limitations of the dynamic algorithm during overloaded conditions. The proposed algorithm D_EDF, simulated and tested for independent, preemptive, periodic tasks on tightly coupled real-time multiprocessor systems under global scheduling. The experiments and result analysis concludes that the proposed algorithm is efficient in both underloaded and overloaded conditions. It always performs better than the conventional EDF algorithm [13].

2.4 The Swarm based Scheduling Algorithms

Any problem solving is a search for the optimal solution from a vast solution space. Artificial Intelligence (AI) solves the problem efficiently by heuristic search, embedding the domain knowledge, which guides the search intelligently. AI has successfully demonstrated its capabilities in almost every field of engineering. Swarm Intelligence (SI) is a computational and behavioral metaphor for problem-solving that originally took its inspiration from nature's examples of collective behaviors like Social Insects: Nest building, Foraging, Assembly, Sorting, and Vertebrates: Swarming, Flocking, Herding, Schooling. SI provides a basis, making it possible to explore collective (or distributed) problem-solving methodology without centralized control [21]. Based on SI, the following scheduling algorithm has been proposed for Soft Real-Time System.

1) ACO Based Scheduling Algorithm - 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. ACO based scheduling algorithm has been compared with the Earliest Deadline First (EDF) algorithm. It is noted that the new algorithm is equally efficient 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, whereas the EDF algorithm performance is degraded in overload condition [11].

3 The STATIC, DYNAMIC, AND HYBRID SCHEDULING ALGORITHMS

This section describes the Researchers contribution to the Static and Dynamic Scheduling Algorithms. It is also explaining the proposed hybrid scheduler, which is using characteristics of Static and Dynamic scheduler. These all schedulers are implemented in a common simulator and environment and compared with Success Ratio (SR) and Effective CPU Utilization Parameter, which is also explained in this section.

3.1 The Simulation Environment and Performance Parameters

3.1.1 The Simulation Environment

The entire simulator for all algorithms has been developed in C programming language, and the compiler is GNU GCC. The simulator has been executed on hardware configuration – Core i5 processor with 8 GB of RAM. Simulation has been carried out on a 64-bit Windows 10 Enterprise operating system. All algorithm has been evaluated with a periodic task set. This simulator is using an extensive data set of periodic tasks [22].

3.1.2 The Performance Parameters

All algorithms' performance has been tested with two primary parameters called SR (Success Ratio) and ECU (Effective CPU Utilization) in this research. These parameters have been described as follows.

1) Success Ratio (SR) - Success Ratio with real-time systems defined as the ratio of a set of processes that meet their deadline and a total number of processes. Success Ration determined with the following equation 1 [23],

$$SR = \frac{Number\ of\ Task\ successfully\ scheduled}{Total\ Number\ of\ Task\ arrived} \qquad (1)$$

2) Effective CPU Utilization (ECU) - Effective CPU Utilization is defined as how much CPU time has been utilizing for the processes which can meet their deadline. ECU determined with the following equation 2 [23],

$$ECU = \sum_{i \in R} \frac{V_i}{T} \qquad (2)$$

Where,

- V represents process value and,
    - Process Value = time required to complete the process if the process meets its deadline.
    - Process Value = 0 if the process does not meet the deadline.
- R is a set of processes, which are scheduled successfully, i.e., completed within their deadline.
- T is the total time of scheduling.

3.2 Critical Analysis of the Static and Dynamic Scheduling Algorithms

3.2.1 Research Carried Out

Researchers have implemented the two dynamic scheduling algorithms (EDF and LST) and two static algorithms (RM and SJF) for the soft RTOS. Algorithms are tested with a periodic task set (describe in section 3), and results are collected. It has observed the success ratio (SR) and effective CPU utilization (ECU) for all algorithms in a similar environment (describe in section 3). It has been observed that the EDF and LST (dynamic algorithms) perform well in underload conditions, but in an overload situation, they cannot perform well. In contrast, the RM and SJF (static algorithms) fail to schedule a specific process in the underload scenario. They perform well in an overload situation compared with the dynamic algorithm [23][24][25][26].

3.2.2 Result and Analysis

EDF, LST, RM, and SJF algorithms are implemented and evaluated with SR and ECU parameters, and results have been gathered. All algorithms have been assessed in underload and overload situation wherein underload $U_p \leq 1$ and in overload $U_p > 1$. Observation with these results indicates that ECU values persist nearly the same for Dynamic and Static algorithms, but SR values are not 100% with the Static scheduling algorithms. When $U_p \leq 1$, it indicates that scheduling a given task set is possible, but Static scheduling algorithms fail to schedule all processes, whereas the dynamic scheduling algorithm can schedule this process set. Dynamic scheduling algorithms give optimum results in the underload scenario, and it is advisable to use the Dynamic schedulers with underload conditions. The results of the overload situation and observation indicate that Dynamic algorithms performance reduces quickly. In contrast, Static algorithms like RM and SJF can still meet a few of their deadline for a given process set. This observation can conclude that EDF and LST in underload give optimal results, whereas, in overload, RM and SJF performed well. Figure 1 and Figure 2 represent the all algorithm performance comparison with respect to ECU% and SR%.

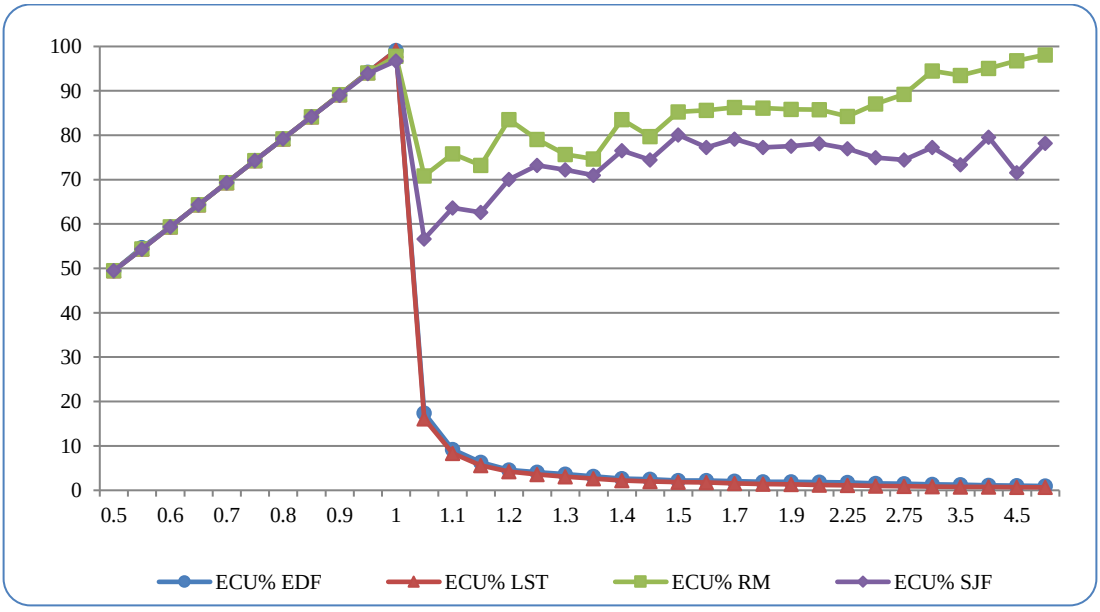

Fig. 1 – ECU% Vs. Load

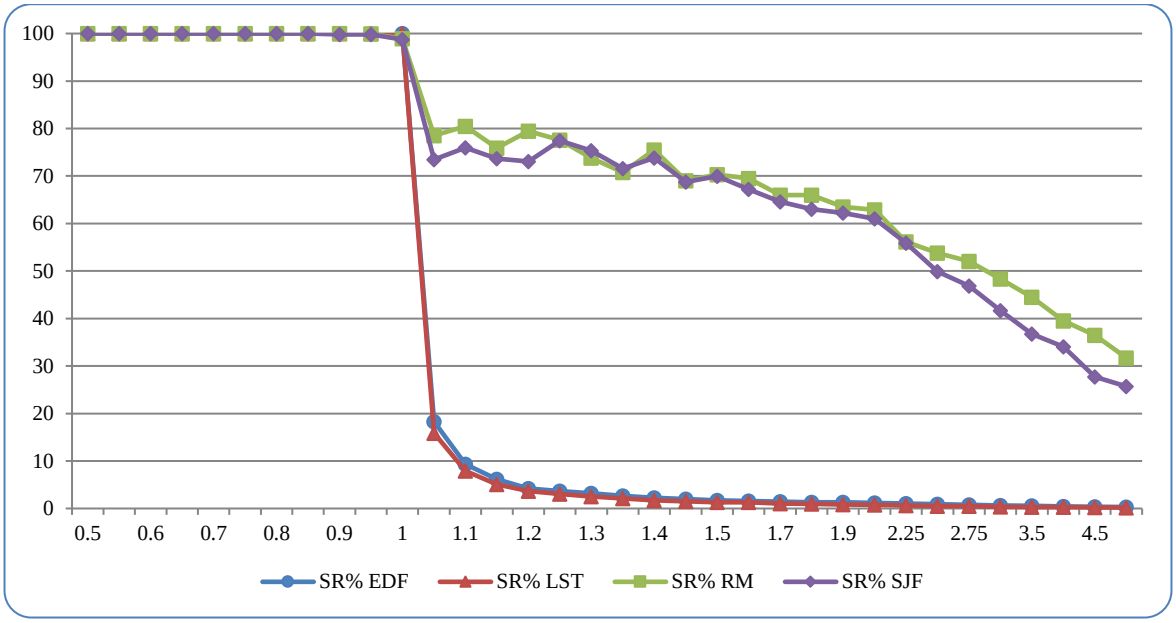

Fig. 2 – SR% Vs. Load

3.3 Hybrid Scheduler (S_LST)

3.3.1 S_LST Algorithm

S_LST algorithm uses the characteristics of LST and SJF. In underload, task priority will be given based on slack time, and in an overload situation, task priority will be assigned based on the shortest execution time. It has been considered that the execution time of the task, its arrival time, its period, and total CPU load is available with Soft Real-Time System. The scheduling algorithm executes when a currently running task completes or a new task arrives. When the scheduling algorithm invokes; first, it observed the CPU load, based on the current process set, and available processes are ready for scheduling. If $U_p < 1$ it will assign the task priority based on slack time (Dynamic scheduling algorithm), and if $U_p > 1$, it will assign the task priority based on the shortest execution time (Static Scheduling algorithm). The static scheduler aims to gain maximum profit from the given process set. So, in an overload situation where the dynamic scheduler performs poorly, the SJF algorithm gets more processes that meet their deadline [27].

3.3.2 Result Analysis and Performance Comparison

S_LST, LST, and SJF algorithms have been implemented on the simulator and evaluated with the dataset mention in section 3. When the CPU load is less than one or equal to one ($U_p \leq 1$), results show that S_LST performs equally well in underload scenarios like the LST algorithm in terms of ECU and SR parameters. S_LST uses the slack time value of the task to assign dynamic priority in the underload situation. When the CPU load is greater than 1 ($U_p > 1$). Results show a waste difference in ECU and SR values compare to a simple LST algorithm. When Load is greater than 1, it means that the task set is not schedulable, and most of the process misses their deadline with the LST algorithm. It has been noted that in slightly overload situations, LST performance degrades very poorly, whereas SJF can meet the deadline for a few of their process sets. It means in an overload situation, SJF gives better performance than LST. That is why S_LST uses static priority in an overload situation. Fig. 3 and Fig. 4 provide a graphical representation of the performance of the S_LST algorithm compare to LST and SJF.

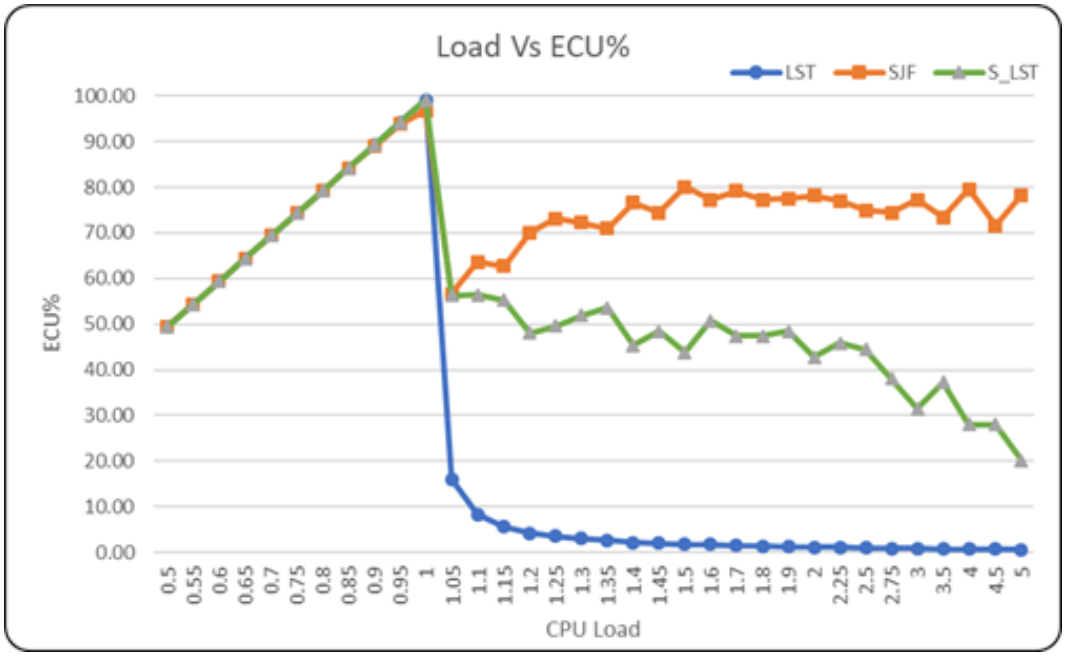

Fig.3 – S_LST, LST and SJF performance (Load Vs. ECU%)

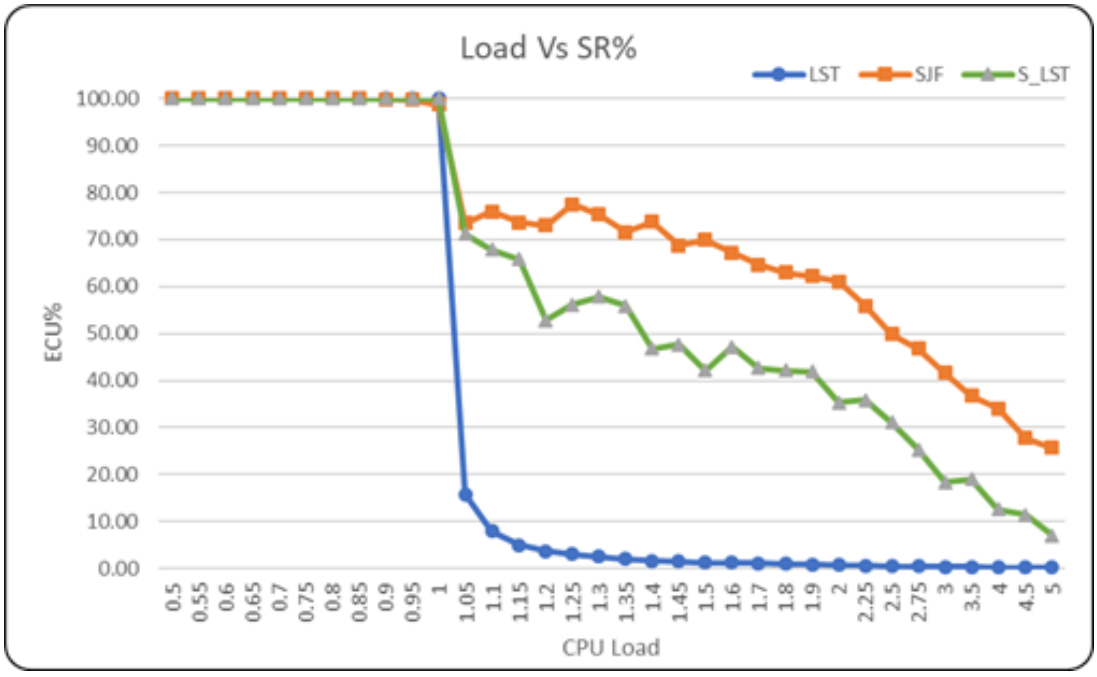

Fig.4 – S_LST, LST and SJF performance (Load Vs. SR%)

4 SWARM INTELLIGENCE AND PROPOSED PSO BASED SCHEDULING ALGORITHM

Swarm intelligence is the study of computational systems inspired by collective intelligence. Collective Intelligence emerges through the cooperation of large numbers of homogeneous agents in the environment. Examples include schools of fish, flocks of birds, and colonies of ants. Such intelligence is decentralized, self-organizing, and distributed throughout an environment. In nature, such systems are commonly used to solve problems such as effective foraging for food, prey evading, or colony re-location. The information is typically stored throughout the participating homogeneous agents or is stored or communicated in the environment itself, such as through the use of pheromones in ants, dancing in bees, and proximity in fish and birds. Particle Swarm Optimization investigates probabilistic algorithms inspired by the flocking, schooling, and herding. Like evolutionary computation, swarm intelligence algorithms or strategies are considered adaptive strategy and are typically applied to search and optimization domains [28].

4.1 Swarm Techniques

Researchers have studied may Swarm Intelligence Techniques to identify the best fitted for problem statement. It is an emerging field of biologically-inspired artificial intelligence based on the behavioral models of social insects such as ants, bees, wasps, termites, etc. The following are the few well-applied Swarm Techniques [29].

1) PSO - Particle Swarm Optimization belongs to Swarm Intelligence and Collective Intelligence and is a Computational Intelligence sub-field. PSO is inspired by the social foraging behavior of some animals, such as birds' flocking behavior and the schooling behavior of fish. Particles in the Swarm fly through an environment following the Swarm fitter members and generally biasing their movement toward historically good areas of their environment. The algorithm's goal is to have all the particles locate the optima in a multi-dimensional hyper-volume [30].

2) ACO - It is inspired by the pheromone communication of the blind ants regarding the right path between the colony and the food source in an environment. The probability of the ant following a particular route is a function of pheromone intensity and a function of distance to that city, the function known as visibility. The strategy's objective is to exploit historical, i.e., pheromone-based and heuristic information to construct candidate solutions each in a probabilistic step-wise manner and fold the information learned from constructing solutions into the history. The probability of selecting a component is determined by the heuristic contribution of the component to the solution's overall cost. The quality of the solution and history is updated proportionally to the quality of the best-known solution [31].

3) Bees - It is inspired by the foraging behavior of the honey bees. The hive sends out the Scout bees when locating nectar (a sugary fluid secreted within flowers), return to the hive, and communicate the other bees the fitness, quality, distance, and direction of the food source via waggle dance. The objective of the algorithm is to locate and explore good sites within a problem search space. Many scout bees are sent out; each iteration is always searching for additional good sites that are continually exploited in the local search application [32].

4) GSA - Gravitational search algorithm (GSA) is a newly developed stochastic search algorithm based on the Newtonian gravity- “Every particle in the universe attracts every other particle with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them” and the mass interactions. In this approach, the search agents are a collection of masses that interact with each other based on the Newtonian gravity and the laws of motion in which all of the objects attract each other by the gravity force. In contrast, this force causes a global movement of all objects towards the objects with heavier masses. The heavy masses correspond to good solutions to the problem [33].

4.2 Ant Colony Optimization

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. Researchers have proposed this algorithm for Soft Real-Time System and practically prove this algorithm's efficiency [11]. Based on mathematical proof, Researchers have again demonstrated the ACO-based scheduling algorithm [34].

4.2.1 ACO based Scheduling Algorithm

The scheduling method must execute when a directly running task completes or any new task gets generated. The method's main steps 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 journey is calculated. We studied this foundation 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.

Algorithm Key Points

1) All schedulable tasks are considered as a node, they store τ values, and it is pheromone. The pheromone τ is initialized with the value 1 for each node.
2) 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.
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 decided based on the number of executable tasks.

4.2.2 Mathematical Proof for ACO based Scheduling Algorithm

The probability of each node will be calculated, and 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.

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

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

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 $P_{max} = 1$.

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

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

4.3 Proposed PSO based Scheduling Algorithm

During the literature study, it has been observed that Particle Swarm Optimization has been widely used in scheduling for the Cloud Computing environment. Researcher A. S. Ajeena Beegom and M. S. Rajasree proposed the Integer-PSO algorithm for task scheduling in a cloud computing system in 2019 [35]. A two-level particle swarm optimization algorithm was created for the flexible job-shop scheduling problem [36]. PSO-based scheduling also applied in workflow applications in Cloud Computing Environments by researchers [37]. The PSO-based scheduling approach has not been explored with the Real-Time Operating system, and researchers in this research have targeted PSO based Swarm Intelligence Techniques to resolve the problem statement. Researchers have selected this approach because it is highly recommended for scheduling problems and it is the right approach when the problem size is between 20 to 40 [38][28].

4.3.1 Particle Swarm Optimization Technique

The Particle Swarm Optimization algorithm is comprised of a collection of particles that move around the search space influenced by their own best past location and the best past location of the whole Swarm or a close neighbour. Each iteration, a particle's velocity is updated using equation 3 [28][39].

$$v_i(t+1) = v_i(t) + \left(c_1 \times rand() \times \left(p_i^{best} - p_i(t)\right)\right) + \left(c_2 \times rand() \times \left(p_{gbest} - p_i(t)\right)\right) \quad (3)$$

Where,

- $v_i(t+1)$ is the new velocity of i^{th}particle,
- c_1 and c_2 are coefficients for the personal best and global best positions respectively,
- $p_i(t)$ is the i^{th} particle's position at time t,
- p_i^{best} is the i^{th} particle's best-known position,
- $p_{gbest}$is the best position known to the Swarm,
- $rand()$function generate a uniformly random variable ∈ [0, 1]

Variants on this update equation consider the best positions within a particle's local neighborhood at time t. A particle's position is updated using the following equation 4 [28].

$$p_i(t+1) = p_i(t) + v_i(t+1) \quad (4)$$

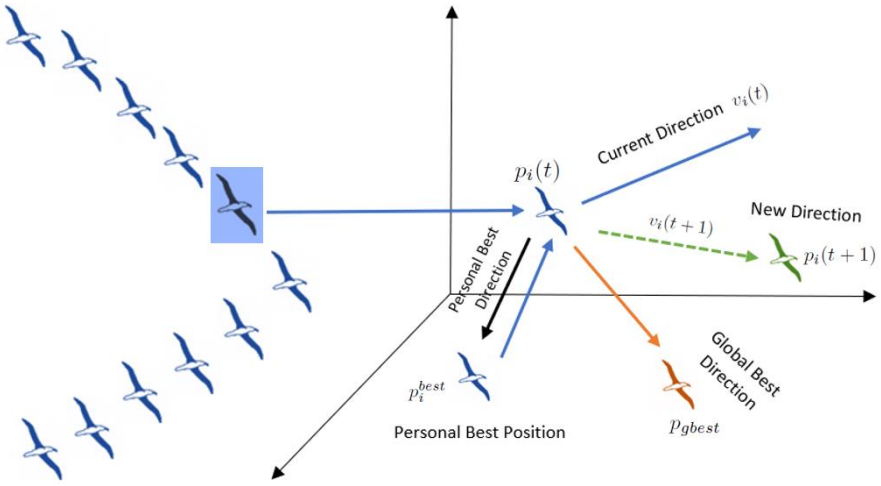

Fig. 5. Graphical representation of PSO

Figure 5 shows the graphical representation of the Particle Swarm Optimisation. After each iteration particle moves in a new direction, which is optimal, and that decision will be based on the personal best position and global best position.

Heuristics for this approach are [38][28][39]:

- The number of particles should be low, around 20-40,
- The speed a particle can move should be bounded,
- The learning factors should be between 0 and 4, typically 2,
- Particles may leave the boundary of the problem space and be penalized, reflected in the domain, or biased to return toward a position in the problem domain. Alternatively, a wrapping strategy may be used at the edge of the domain, creating a loop, torrid, or related geometrical structures at the chosen dimensionality.

An inertia or momentum coefficient can be introduced to limit the change in velocity.

4.3.2 PSO Based Scheduling Algorithm

The algorithm considering each given task as a particle, and all tasks which are eligible for scheduling are viewed as a set of particles. The scheduler's ultimate goal is to choose a task at a given point in time in such a way that the task can meet its deadline. The Soft Real-Time system is intended to make sure that all tasks meet their deadline in the underload condition, and the maximum task will meet their deadline in the overload scenario.

The scheduling algorithm will be executed when a new task arrives, or the currently performing task is completed. When there is more than one task that is ready to run, the scheduler needs to select the task effectively. The PSO based scheduler has the following significant steps.

1) Initialization of Task as a Particle
2) Compute the velocity and position of each task
3) Analyze the position and velocity of each task
4) Selection of Task for execution

4.3.3 Critical Analysis of PSO Based Scheduling Algorithm

The proposed algorithm has been compared with EDF (Earliest Deadline First) and ACO (Ant Colony Optimization) based algorithm. The correctness of all three algorithms has been tested under a similar Hardwar and dataset, as described in section 3. These algorithms have been implemented in the simulator using the C language. These algorithms have been compared with parameter SR and ECU, as described in section 3. Convergence Analysis and Parameter selection of the proposed algorithm is also given by the researchers based on zheng (2003) research [40].

4.3.4 Result and Comparisons

Researchers have compared the practical performance of the proposed algorithm with EDF and ACO based scheduling algorithm. Performance parameters ECU% and SR% have been taken into consideration. It has been noted that when CPU load is less than or equal to one ($U_p \leq 1$), ACO and EDF algorithm make sure to meet all the given task set deadlines. Whereas PSO based scheduler misses a few deadlines in a specific case. In an overload situation when CPU load is more than one ($U_p > 1$), EDF performance degrades rapidly; even ACO can perform better than EDF and try to meet the maximum deadline. PSO based scheduler even performs better than ACO based scheduler in an overload condition. Figure 6 and Figure 7 show the performance of EDF, ACO, and PSO based scheduler with respect to ECU and SR parameter.

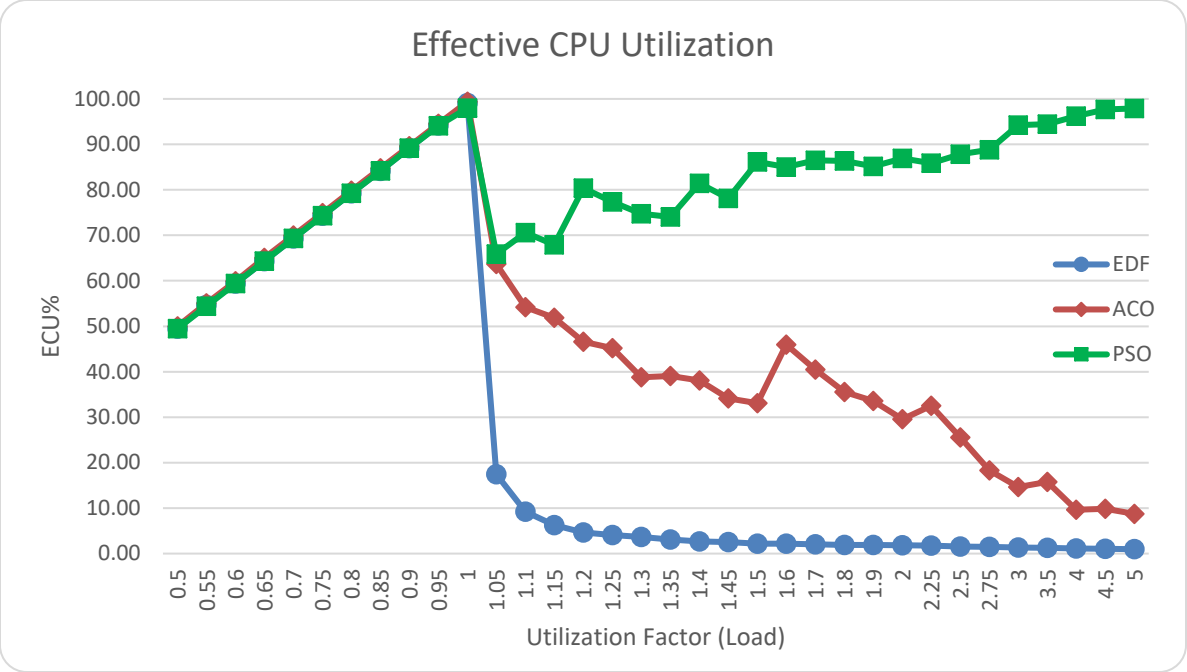

Fig. 6 - ECU% comparison of EDF Vs. ACO Vs. PSO Algorithm

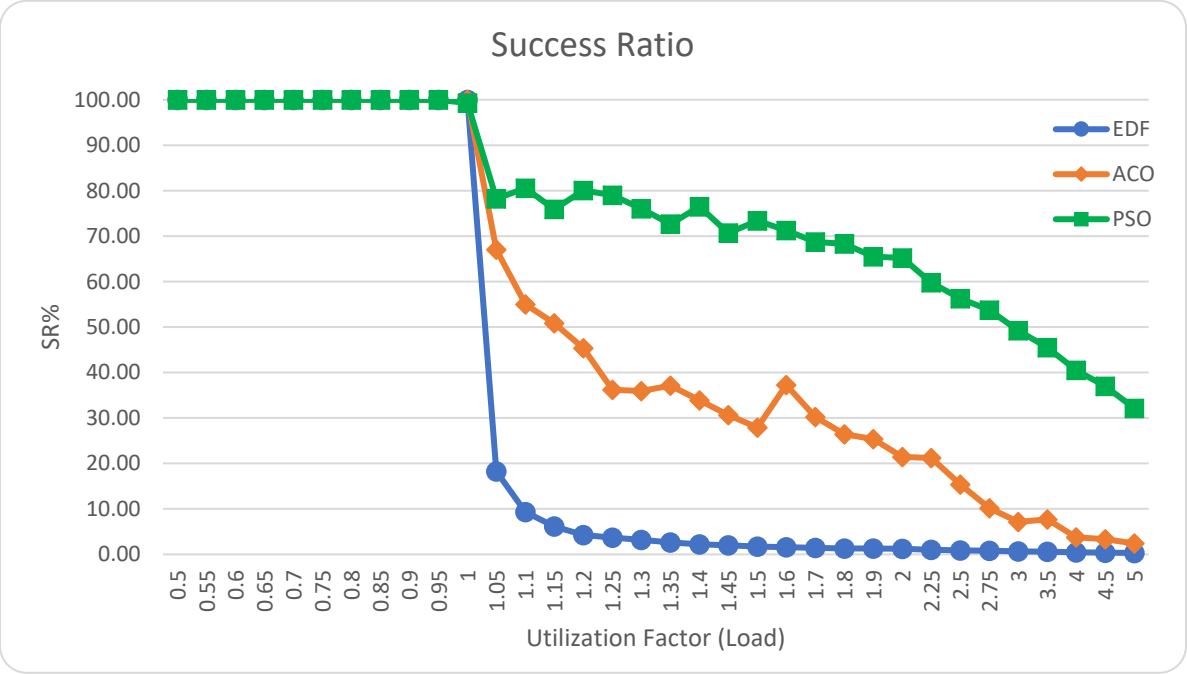

Fig. 7 - ECU% comparison of EDF Vs. ACO Vs. PSO Algorithm

4.3.5 Conclusion and Road Map for Future Work

The proposed PSO based scheduling algorithm has been compared with EDF and ACO based scheduling algorithm under the Soft Real-Time periodic task set. The performance parameters SR and ECU have been calculated for each algorithm for a large dataset, and a comparison has been made. It has been observed that during the underload scenario ($U_p \leq 1$) proposed scheduling algorithm performs similar to the EDF and ACO based algorithm. In a slightly overload situation when $1.00 \leq U_p \leq 1.5$, EDF performance gets degraded sharply. The proposed algorithm and ACO based scheduling algorithm performs better than EDF, and even the proposed approach delivers better than the ACO based scheduling algorithm. During highly overload scenario ($1.50 \leq U_p \leq 5.00$) EDF and ACO-based algorithms perform poorly, whereas the PSO-based scheduling algorithm can still meet a specific deadline. So, instead of static or dynamic priority, the proposed approach works well during underload, overload, and highly overload scenarios.

The proposed method is tested with uniprocessor and periodic task set for Soft Real-Time System. In future work, this algorithm can be examined with Hard and Firm Real-Time System as well. By making a few changes, the modified PSO based scheduling algorithm can be implemented for the multiprocessor system.

5 REFERENCES

[1] S. Ahmad, S. Malik, and D. H. Kim, “Comparative analysis of simulation tools with visualization based on real-time task scheduling algorithms for iot embedded applications,” *Int. J. Grid Distrib. Comput.*, vol. 11, no. 2, 2018.

[2] Y. Laalaoui and N. Bouguila, “Pre-run-time scheduling in real-time systems: Current researches and Artificial Intelligence perspectives,” *Expert Syst. Appl.*, vol. 41, no. 5, pp. 2196–2210, 2014.

[3] K. Chatterjee, A. Pavlogiannis, A. Kößler, and U. Schmid, “Automated competitive analysis of real-time scheduling with graph games,” *Real-Time Syst.*, vol. 54, no. 1, pp. 166–207, 2018.

[4] A. Magdich, Y. Hadj Kacem, M. Kerboeuf, A. Mahfoudhi, and M. Abid, “A design pattern-based approach for automatic choice of semi-partitioned and global scheduling algorithms,” *Inf. Softw. Technol.*, vol. 97, no. November 2017, pp. 83–98, 2018.

[5] J. Teraiya and A. Shah, “Comparative Study of LST and SJF Scheduling Algorithm in Soft Real-Time System with its Implementation and Analysis,” *2018 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2018*, pp. 706–711, 2018.

[6] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment,” *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[7] G. Saini, “Application of fuzzy logic to real-time scheduling,” in *2005 14TH IEEE-NPSS Real Time Conference*, 2005.

[8] S. Baruah *et al.*, “On the competitiveness of on-line real-time task scheduling,” *Real-Time Syst.*, 1992.

[9] S. C. Yu, “Elucidating multiprocessors flow shop scheduling with dependent setup times using a twin particle swarm optimization,” *Appl. Soft Comput. J.*, vol. 21, pp. 578–589, 2014.

[10] H. Kazemi, Z. M. Zahedi, and M. Shokouhifar, “Swarm Intelligence Scheduling of Soft Real-Time Tasks in Heterogeneous Multiprocessor Systems,” *Electr. Comput. Eng. An Int. J.*, vol. 5, no. 1, 2016.

[11] A. Shah and K. Kotecha, “Scheduling algorithm for real-time operating systems using ACO,” in *Proceedings - 2010 International Conference on Computational Intelligence and Communication Networks, CICN 2010*, 2010.

[12] F. Lindh, T. Otnes, and J. Wennerström, “Scheduling algorithms for real-time systems,” *Dep. Comput. Eng. Malardalens Univ. Sweden*, 2010.

[13] D. Thakor and A. Shah, “D_EDF: An efficient scheduling algorithm for real-time multiprocessor

system," *Inf. Commun. Technol. (WICT), 2011 World Congr.*, pp. 1044–1049, 2011.

[14] G. Koren and D. Shasha, "Dover: an optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems," *SIAM J. Comput.*, 1995.

[15] D. G. Harkut, "Comparison of Different Task Scheduling Algorithms in RTOS : A Survey," vol. 4, no. 7, pp. 1236–1240, 2014.

[16] G. C. Buttazzo, "Rate Monotonic vs. EDF: Judgment day," *Real-Time Syst.*, vol. 29, no. 1, pp. 5–26, 2005.

[17] M. Patel and B. Oza, "International Journal of Advance Engineering and Research An Improved LLF _ DM Scheduling Algorithm for Periodic Tasks by Reducing Context Switches," vol. 0, pp. 248–254, 2015.

[18] K. Ramamritham, J. A. Stankovic, and P. F. Shiah, "Efficient Scheduling Algorithms for Realtime Multiprocessor Systems," *IEEE Trans. Parallel Distrib. Syst.*, 1990.

[19] W. Li, K. Kavi, and R. Akl, "A non-preemptive scheduling algorithm for soft real-time systems," *Comput. Electr. Eng.*, vol. 33, no. 1, pp. 12–29, 2007.

[20] Shah Apurva, "Dynamic Scheduling for Real-Time Operating Systems," pp. 1–141, 2009.

[21] A. Shah, "Adaptive scheduling for real-time distributed systems," *Biol. Tech. Knowl. Discov. Data Min.*, pp. 236–248, 2014.

[22] "Real Time System Dataset." [Online]. Available: http://www.processdataset.in/.

[23] J. Teraiya and A. Shah, "Analysis of Dynamic and Static Scheduling Algorithms in Soft Real-Time System with Its Implementation," *Adv. Intell. Syst. Comput.*, vol. 1053, pp. 757–768, 2020.

[24] R. Belagali, S. Kulkarni, V. Hegde, and G. Mishra, "Implementation and validation of dynamic scheduler based on LST on FreeRTOS," *2016 Int. Conf. Electr. Electron. Commun. Comput. Optim. Tech. ICEECCOT 2016*, pp. 325–330, 2017.

[25] G. Chen and W. Xie, "On a laxity-based real-time scheduling policy for fixed-priority tasks and its non-utilization bound," *2011 Int. Conf. Inf. Sci. Technol. ICIST 2011*, pp. 7–10, 2011.

[26] T. Feld, A. Biondi, R. I. Davis, G. Buttazzo, and F. Slomka, "A survey of schedulability analysis techniques for rate-dependent tasks," *J. Syst. Softw.*, vol. 138, pp. 100–107, 2018.

[27] J. Teraiya and A. Shah, "Hybrid Scheduler (S_LST) for Soft Real-Time System based on Static and Dynamic Algorithm," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 2, pp. 2885–2889, 2019.

[28] J. Brownlee, *Clever Algorithms*. 2011.

[29] "Swarm Intelligence." [Online]. Available: http://www.techferry.com/articles/swarm-intelligence.html.

[30] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *IEEE International Conference on Neural Networks - Conference Proceedings*, 1995.

[31] M. Dorigo, G. Di Caro, and L. M. Gambardella, “Ant algorithms for discrete optimization,” *Artif. Life*, 1999.

[32] B. Yuce, M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase, “Honey bees inspired optimization method: The bees algorithm,” *Insects*, vol. 4, no. 4, pp. 646–662, Nov. 2013.

[33] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, “GSA: A Gravitational Search Algorithm,” *Inf. Sci. (Ny).*, 2009.

[34] J. Teraiya, A. Shah, and K. Kotecha, “ACO based scheduling method for soft RTOS with simulation and mathematical proofs,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12, pp. 4736–4740, 2019.

[35] A. S. A. Beegom and M. S. Rajasree, “Integer-PSO: a discrete PSO algorithm for task scheduling in cloud computing systems,” *Evol. Intell.*, vol. 12, no. 2, pp. 227–239, 2019.

[36] R. Zarrouk, I. E. Bennour, and A. Jemai, “A two-level particle swarm optimization algorithm for the flexible job shop scheduling problem,” *Swarm Intell.*, vol. 13, no. 2, pp. 145–168, 2019.

[37] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, “A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments,” *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 400–407, 2010.

[38] R. Eberhart and J. Kennedy, “New optimizer using particle swarm theory,” in *Proceedings of the International Symposium on Micro Machine and Human Science*, 1995.

[39] M. Elbes, S. Alzubi, T. Kanan, A. Al-Fuqaha, and B. Hawashin, “A survey on particle swarm optimization with emphasis on engineering and network applications,” *Evol. Intell.*, vol. 12, no. 2, pp. 113–129, 2019.

[40] Y. L. Zheng, L. H. Ma, L. Y. Zhang, and J. X. Qian, “On the convergence analysis and parameter selection in particle swarm optimization,” *Int. Conf. Mach. Learn. Cybern.*, vol. 3, no. November, pp. 1802–1807, 2003.

6 PUBLICATIONS

[1] Jay Teraiya, Apurva Shah, *"Comparative Study of LST and SJF Scheduling Algorithm in Soft Real-Time System with its Implementation & Analysis",* In the proceedings of IEEE ICACCI-2018, PES Institute of Technology, Bangalore, India (Indexed By Web of Science, SCOPUS, DBLP)

[2] Jay Teraiya, Apurva Shah*, "Analysis of EDF and RM Scheduling Algorithm in Soft Real-Time Operating System",* this paper has been submitted Journal of the Maharaja Sayajirao University of Baroda. (Indexed By UGC Care)

[3] Jay Teraiya, Apurva Shah, *"Analysis of Dynamic and Static Scheduling Algorithms in Soft Real-Time System with its Implementation"*, Advances in Intelligent Systems and Computing (AISC) Series, Springer, (Proceedings of SoCTA-2018) (Indexed by DBLP, SCOPUS, SpringerLink)

[4] Jay Teraiya, Apurva *Shah "Hybrid Scheduler (S_LST) for Soft Real-Time System based on Static and Dynamic Algorithm*" International Journal of Engineering and Advanced Technology, ISSN: 2249 – 8958, Volume-9, Issue-2 December 2019.

[5] Jay Teraiya, Apurva Shah, Ketan Kotecha *"ACO Based Scheduling Method for Soft RTOS with Simulation and Mathematical Proofs",* International Journal of Innovative Technology and Exploring Engineering, ISSN: 2278-3075, Pg. no. 4736-4740, Volume-8, Issue-12, October 2019. (Indexed by SCOPUS)

[6] Jay Teraiya, Apurva Shah *"Optimized Scheduling Algorithm for Soft Real-Time System using Particle Swarm Optimization Technique",* paper has been submitted to Evolutionary Intelligence Journal of Springer and it is under 2nd review with minor revision.