STM JOURNALS

Journal of
Operating Systems Development & Trends

ISSN: 2454-9355
Volume 8, Issue 2, 2021
DOI (Journal): 10.37591/JoOSDT

http://computers.stmjournals.com/index.php?journal=JoOSDT&page=index

Research JoOSDT

# An Optimised CPU Scheduling Algorithm with Adaptive Time Quantum Approach

Pradyut Nath[1,*], Sumagna Dey[1], Srija Nandi[1], Subhrapratim Nath[2]

***Abstract***

*CPU scheduling is an essential mechanism implemented by the operating system to determine the execution of multiple processes by the CPU. The primary objective of the scheduling algorithms is to optimize the systems' performance efficiently. The performance of a CPU scheduling algorithm depends on various factors and can be evaluated on various criteria like average turnaround time, average waiting time, throughput, fairness etc. This paper aims to present an optimal CPU scheduling algorithm, Adaptive Quantum Round Robin (AQRR) in a uniprocessor environment. Related work has been done on increasing the performance of existing Round Robin Algorithms using dynamic time quantum approaches, but the maximum percentage gain in turnaround time and waiting time using these approaches, over the traditional Round Robin algorithm is not more than 30% and 40% respectively. The proposed algorithm in this paper outperforms these algorithms in both turnaround time and waiting time. The AQRR algorithm is based on the standard Round Robin algorithm, but it is integrated with a dynamic time quantum which is self-adaptive triggered on the event of the arrival of new processes in the ready queue or the completion of a process in the process queue. The dynamic time quantum is updated based on the remaining burst time of the running processes in the process queue at that instance as well as a weightage associated with it. Finally, a comparative study between the prevailing similar scheduling algorithms and the proposed algorithm is observed and noted, based on different scenarios. The following algorithms are compared on two criteria: average turnaround time and average waiting time.*

**Keywords:** Operating system, scheduling algorithms, weighted average, round robin, resource management

## INTRODUCTION

The operating system [1] acts as an interface between hardware and software. It manages memory allocation and deallocation, allocates processors to processes, and handles IO devices and file resources. As a form of parallel processing, the processes run at the same time on a uniprocessor in multiprogramming, to maximize CPU utilization. Since there is a single processor, no real simultaneous execution occurs, rather the OS executes the processes' part by part in a cyclic fashion.

Every process from creation generally can be divided into the following stages: new, ready, running, waiting, and termination state, as shown in Figure 1. In the new stage, the program is created. The long-term scheduler [2] then loads the new process to the ready queue present in the main

***Author for Correspondence**
Pradyut Nath
E-mail: pradyutnathradhae@gmail.com

[1]Students, Department of Computer Science and Engineering, Meghnad Saha Institute of Technology, Kolkata, West Bengal, India
[2]Department Head and Assistant Professor, Department of Computer Science and Engineering, Meghnad Saha Institute of Technology, Kolkata, West Bengal, India

Received Date: July 01, 2021
Accepted Date: July 27, 2021
Published Date: August 10, 2021

**Citation:** Pradyut Nath, Sumagna Dey, Srija Nandi, Subhrapratim Nath. An Optimised CPU Scheduling Algorithm with Adaptive Time Quantum Approach. Journal of Operating Systems Development & Trends. 2021; 8(2): 1–18p.

© STM Journals 2021. All Rights Reserved 

memory. After that, the short-term scheduler [2] loads the process from the ready queue to the processor. When the process has the processor allocated to it, it is said to be in the running state. A running process may become suspended if it makes an IO request. Medium-term scheduler [2] in such scenarios removes the process from the main memory to the waiting queue. The information about the existing process is stored in the process control block initialized by the operating system during the process creation. After a process finishes its execution, it will be terminated, and all context of the process created by OS (Process Control Block) will also be deleted [3].

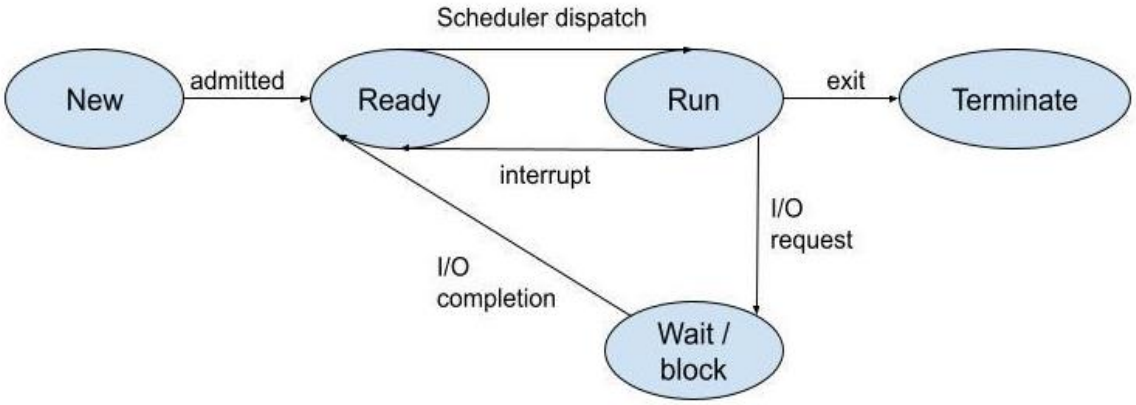



**Figure 1.** Stages of processes in CPU scheduling.

CPU scheduling [4] is an important part of the operating system. Scheduling is the task of selecting which process from the ready queue will be executed in CPU when another process is ON HOLD due to resource wait like I/O. Thus, CPU scheduling aims to make full use of the CPU and reduce CPU idle time. Scheduling algorithms are classified as either preemptive or non-preemptive. In computing, preemption refers to the act of temporarily interrupting the execution of a task, intending to resume it in the future. The scheduling algorithm which uses preemption is called preemptive scheduling. In non-preemptive scheduling, processes cannot be interrupted during their execution. The efficacy of a scheduling algorithm is estimated by preliminary factors such as the CPU utilization time, average waiting time and turnaround time, etc.

In this proposed work, a custom scheduling algorithm is proposed. The above algorithm is based on the classical Round Robin algorithm, but instead of a static time quantum, a dynamic quantum is planned. The dynamic time quanta update itself automatically whenever a new process enters the process queue or an old process terminates, based on a mathematical formulation. Overall, this paper selects an optimal quantum that will self-update itself based on the present condition of processes and their priorities to further reduce starvation and enable the fast execution of prioritized processes.

## RELATED WORKS

Neetu Goel and Dr. R.B. Garg in their work [5] prepared a state diagram that displays the comparative analysis of existing scheduling algorithms on a single CPU and detects the best algorithms in different situations. Their work summarizes and analyzes the efficacy of various algorithms and seeks to develop an efficient CPU scheduler that designs high-quality scheduling algorithms, matching scheduling goals.

In their work, R. Patel and M. Patel proposed a new preemptive CPU scheduling algorithm [6] named SJRR CPU scheduling algorithm, which relies upon the arrival time of a process. This algorithm is based on the traditional round-robin concept. The algorithm concentrates on its preemptive nature and provides a fair CPU execution time based on the average waiting time and turnaround time for a particular process, thus reducing maximum latency.

© STM Journals 2021. All Rights Reserved 

A.R. Dash et al. also proposed a new variant of the Round Robin algorithm in their work [7]. It was found that by replacing the static time quantum of traditional round-robin with a dynamic time quantum as proposed in the new algorithm DABRR (Dynamic Average Burst Round Robin), better performance in terms of average turnaround time, average waiting time, and context switching was obtained.

Zhimin Fang in his work [8] introduced a weight-based multi-objective genetic algorithm to solve the flow shop scheduling problem. The proposed algorithm is a new variant of the weight-based multi-objective immune genetic algorithm (WBMOIGA), differing in two aspects. Firstly, the use of truncation algorithms with similar individuals (TASI) and secondly the modified local search algorithm. This new algorithm is not only able to solve the multi-objective optimization problem with multivariable but also shows better performance than other genetic algorithms like NSGA-II and RWGA.

Real-time processor scheduling for sporadic tasks without missing any tasks and scheduling them within their deadline is a challenging problem in itself. *N.B.* Arune kumar et al. in their work [9] proposed a hybrid bat-inspired algorithm that attempts to obtain the best task allocation schedule for both periodic and sporadic tasks in a soft real-time system. The experiment shows that in several instances balancing both maximum processor utilization and minimized deadline miss, the tasks are scheduled optimally.

## PRELIMINARIES

### Efficiency Parameters of Scheduling Algorithms

The essential criteria that are considered to examine the efficiency of a certain CPU scheduling algorithm [10] are as follows:

*CPU utilization:* Efficient utilization of CPU time and its resources.

*Arrival time:* The time at which a process enters the ready queue.

*Burst time:* The time required by a process to completely execute on CPU.

*Turnaround time:* The total amount of time spent by a process in the CPU from its arrival time till its completion time.

*Waiting time:* The total amount of time spent by the process in the ready queue waiting for the CPU.

*Response time:* The total time spent by a process in the ready queue till its first execution on the CPU.

*Throughput:* The total number of processes completed per unit time.

*Fairness:* Every process should get its fair share of time.

*Priorities:* Processes with higher priority should be executed first and the processes having the same priority should be treated based on first come first serve.

### Weighted Average

The weighted average [11] is a computational technique, where the new value not only depends on its current value but also depends on its previously calculated values.

© STM Journals 2021. All Rights Reserved 

Like tomorrow's weather prediction not only depends on all the weather conditions of today but also depends on the weather forecast of the last seven days. Typical weighted average equation mentioned below where W is the Weighted average and $w_i$, $X_i$ are weight and value of an ith instance of the X variable.

$$W = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i} \quad (1)$$

There are mainly two types of weighted average, e.g. simple weighted average & moving weighted average.

The simple weighted average is a kind of weighted average where it puts equal weight and preference to all previous values. While in moving weighted average, more weight and preference is put on recent data and lesser weight on the past data.

**CPU Scheduling Algorithms**

RR—The Round-robin scheduling algorithm [12] originated from the concept of the round-robin principle in which each person gets an equal share of a certain substance for a certain amount of time. This approach is the most effective way used for multitasking and a time-sharing environment.

Each process is executed for a specific interval of time known as time quantum in cyclic nature. Each process receives a fair share of execution which makes this algorithm starvation-free. The RR scheduling algorithm is preemptive. After each time slice, the CPU switches to the next process, and the processes are preempted to the end of the ready queue. If a new process arrives then it is added to the end of the ready queue and when a process is terminated or it's in a waiting state, then the first process from the queue is allotted to CPU for execution. However, if the time-slicing is low then that affects the output of the processor adversely. In terms of average response time, the RR scheduling algorithm gives the best performance, among others.

The priority-based—Priority scheduling algorithm [13] is non-preemptive and is often used in batch operating systems. The algorithm depends on the arrival time of each process. The processes are sorted on the arrival time and if two processes arrive at the same time then the process with higher priority is executed first. This algorithm follows the principle of first come first serve (FCFS). Furthermore, processes with similar arrival times and priorities are sorted based on the process number and the process with the least process number gets executed first.

SJF—shortest job first (SJF) [13] is probably one of the optimal scheduling algorithms in which a process with minimum burst time is executed first. This scheduling algorithm can be both preemptive and non-preemptive. This algorithm significantly reduces the average waiting time and is often used in batch processing systems. In non-preemptive SJF, each process is allocated to the CPU for its execution as a whole until the process is terminated or in the waiting state. In preemptive SJF, a process with the shortest execution time is allocated for the CPU cycle but when a new process with a shorter burst time arrives, then the executing process is preempted, and the new process is allotted for CPU execution. Therefore, this scheduling algorithm can raise the starvation for a process with a long burst time.

**OUR PROPOSAL**

The proposed algorithm explained below, is based on the classical Round Robin algorithm. The time quantum of this algorithm is designed in such a fashion that itself adjusts itself based on the change of Process Queue. Depending on whether a process has completed execution or the addition of a new process, the self-update module of time quantum is called which changes time-quantum. Finally, when the process queue becomes empty the scheduling algorithm stops.

The algorithm is elaborated below in the next section.

© STM Journals 2021. All Rights Reserved 

**AQRR Algorithm**

T: Timer = 0
TQ: Time Quantum = 0
PQ: Ready Queue
ind: index directing the current executing process of PQ
n: number of processes
PL: Input process List
AL: Arrival List of Processes
Pi: Process at ith index
curr: Index of currently executing process in PQ = 0

1. If all processes arrived in ready queue at same time
2. Add Processes to PQ
3. Arrange the processes in PQ in ascending order of their burst_time.
4. Else
5. Pass
6. Update the TQ
7. Repeat from Step 8 to 17 till (PQ is not empty or ind != length of AL)
8. if T > AL[ind] then
9. Add Pind process from PL to PQ
10. Update TQ
11. curr = curr + 1
12. Execute Pcurr from PQ
13. T = T+ Executed_Time of Pcurr
14. if Pcurr has Zero remaining burst_time
15. Remove Pcurr process from PQ
16. Update TQ
17. ind = ind + 1
18. Stop

**Illustration**

In this section, the proposed algorithm is analyzed and explained as in Figure 2.

In every test case, the number of processes and the respective process's burst time, arrival time, and priority is entered into the simulation.

If all processes have the same arrival time, the processes are all added to process_queue at the same time and the Time quantum is updated based on the average of total burst time as shown in Figure 3(a). Then the processes in process_queue are sorted based on their burst_time in ascending order (if they have the same priority) or are sorted based on their priority in descending order (if they have different priority) [14].

If the processes have different arrival times, the processes are added to process_queue according to their arrival time. Whenever a process is added or removed from process_queue the time quantum is updated according to Figure 3(b).

In all the scenarios, the basic principle of round robin is followed where a variable named curr keeps track of the currently executing process in process_queue and executes the process for the time quantum allotted at that time.

## EXPERIMENTAL ANALYSIS

First, the problem is divided into two main categories—processes with the same priority and processes with different priorities.

© STM Journals 2021. All Rights Reserved 

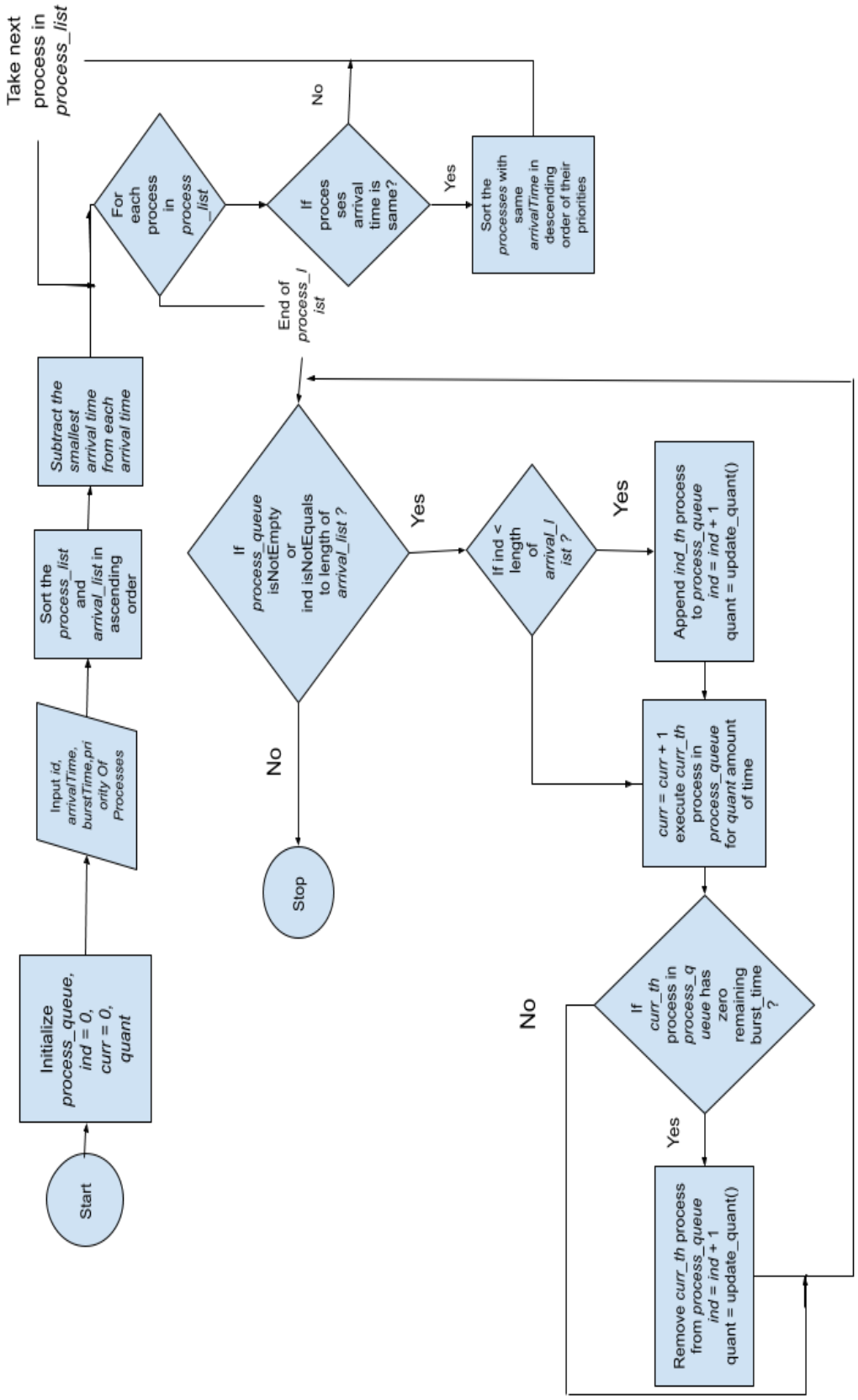



**Figure 2.** Flowchart of Proposed Algorithm.

© STM Journals 2021. All Rights Reserved 

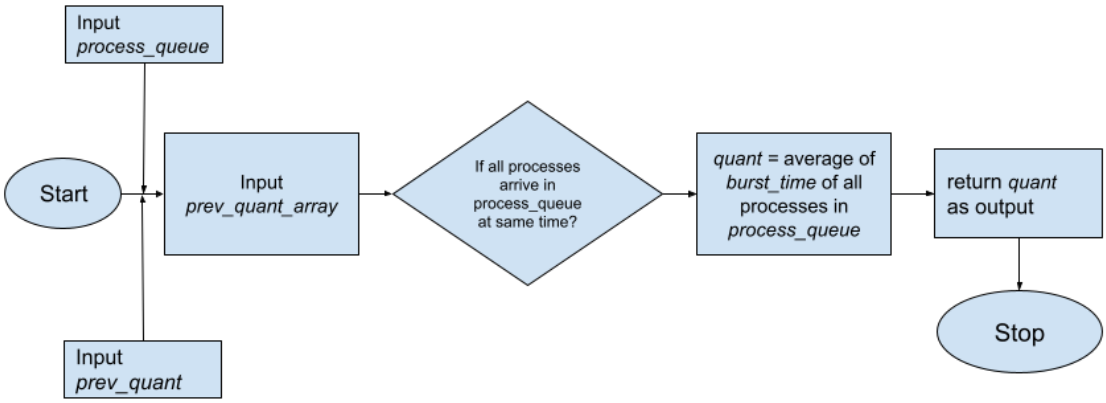



**Figure 3.** (a) Flowchart of quantum update: Case 1.

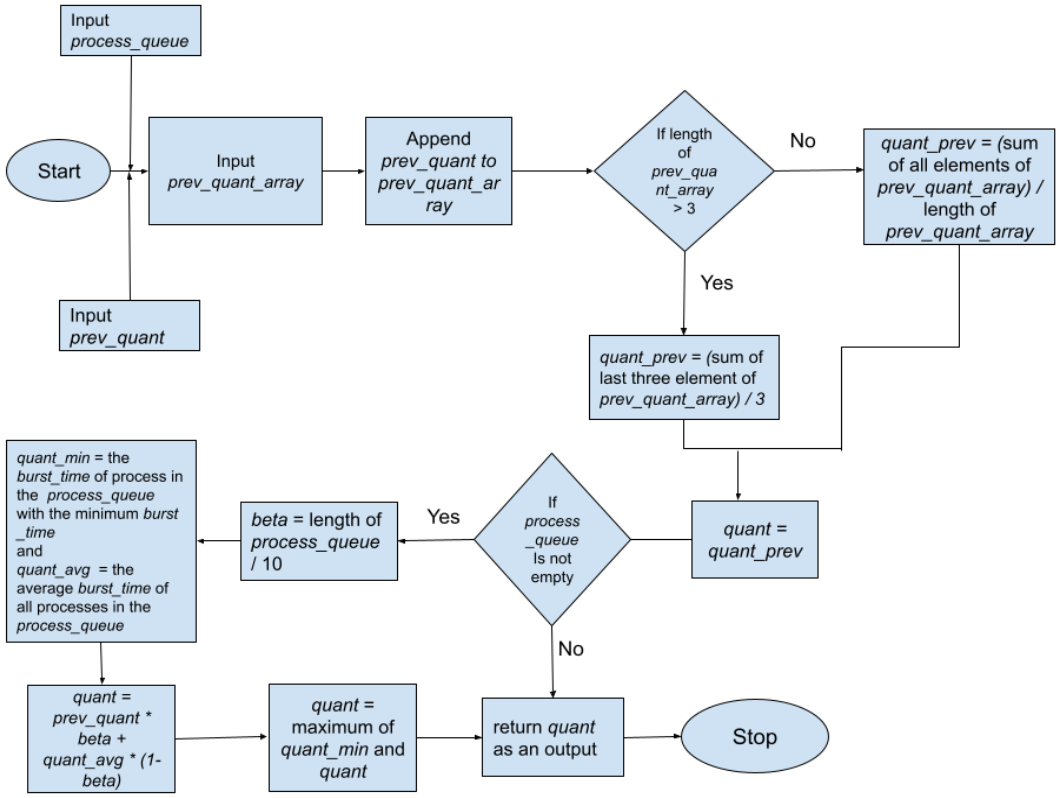



**Figure 3**. (b) Flowchart of quantum update: Case 2.

In the first case where processes with the same priorities are taken, the cases are subdivided based on the arrival time of processes (Zero arrival time processes and processes with different arrival times). Later on, each is further subdivided into three types based on the burst time of processes(In ascending order, in descending order, and in random order). This gives rise to a total of six situations against which the proposed algorithm is compared with four other algorithms-Round-Robin [12], SARR [15], DQRRR [16], MRR [17].

In the second case where processes with different priorities are taken, the work is analyzed against a random set of processes with different priority and Zero arrival times and compared with other algorithms—Round-Robin, priority-based scheduling.

© STM Journals 2021. All Rights Reserved 

In each test scenario, five processes with their process_id, arrival time, burst time, and priority are entered into the system for analysis. Throughout the paper for the Round-Robin algorithm, a static time quantum of 25 is taken for the experiment.

**Assumptions**

During this experiment, only CPU-bound processes are considered. In each test case, five different processes are considered for scheduling in a uniprocessor environment. A simulation environment is created where the burst time, arrival time, priority of the processes is provided by the user at first. Then the simulation environment schedules the processes, simulating a uniprocessor environment. The context switching delay of the processes is considered zero in the given experiment.

**Simulation Environment**

For this system simulation, the whole project is run locally on a desktop with the following specifications as shown in Table 1.

**Table 1.** Simulation Environment.

| **Resource** | **Resource value** |
|---|---|
| **Processor** | I5–7th generation |
| Operating system | Windows 10 |
| Programming language | Python 3 |
| RAM | 8 GB |
| Library used | Numpy |

**With Zero Arrival Time, Same Priority**

***In Ascending Order***

In the following subsection processes with zero arrival time and the same priorities with increasing burst time are taken for comparison (Tables 2-17).

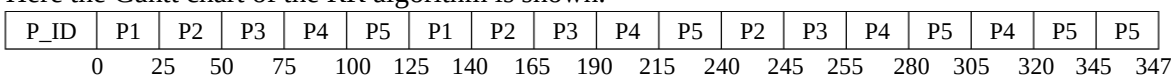
**Table 2.** Processes with Zero arrival time and burst time in ascending order.

| **Processes** | **Arrival Time** | **Burst Time** | **Priority** |
|---|---|---|---|
| P1 | 0 | 40 | 1 |
| P2 | 0 | 55 | 1 |
| P3 | 0 | 60 | 1 |
| P4 | 0 | 90 | 1 |
| P5 | 0 | 102 | 1 |

***Gantt Charts of Algorithms are given below***

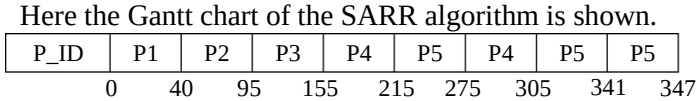
***RR Algorithm***

Here the Gantt chart of the RR algorithm is shown.



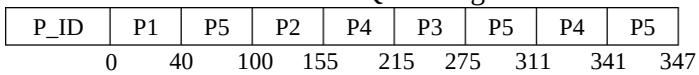

***SARR Algorithm***

Here the Gantt chart of the SARR algorithm is shown.




***DQRRR Algorithm***

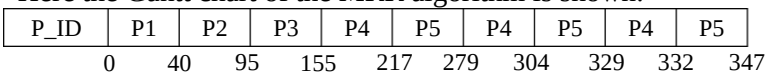
Here the Gantt chart of the DQRRR algorithm is shown.




© STM Journals 2021. All Rights Reserved 

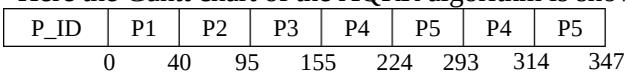
***MRR Algorithm***

Here the Gantt chart of the MRR algorithm is shown.




***AQRR Algorithm***

Here the Gantt chart of the AQRR algorithm is shown.




Average turnround time = (40+95+155+314+347)/5 = 190.2
Average waiting time = (0+ 40+95+224+245)/5 = 120.8

**Table 3.** Comparative study of different algorithms with zero arrival time and burst time in ascending order.

| Algorithm | Avg. waiting time | Avg. turnaround time |
|---|---|---|
| Round-Robin | 192 | 261.4 |
| SARR | 119 | 188.4 |
| DQRRR | 162.2 | 231.6 |
| MRR | 124.4 | 193.8 |
| AQRR | 120.8 | 190.2 |

***In Descending Order***

In the following subsection processes with zero arrival time and the same priorities with decreasing burst time are taken for comparison.

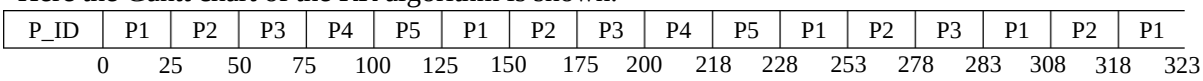
**Table 4.** Processes with zero arrival time and burst time in descending order.

| Processes | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 105 | 1 |
| P2 | 0 | 85 | 1 |
| P3 | 0 | 55 | 1 |
| P4 | 0 | 43 | 1 |
| P5 | 0 | 35 | 1 |

***Gantt Charts of Algorithms are given below***

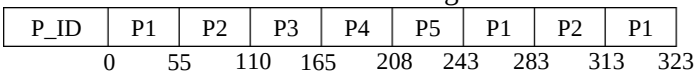
***RR Algorithm***

Here the Gantt chart of the RR algorithm is shown.



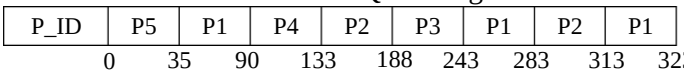

***SARR Algorithm***

Here the Gantt chart of the SARR algorithm is shown.




***DQRRR Algorithm***

Here the Gantt chart of the DQRRR algorithm is shown.




***MRR Algorithm***

Here the Gantt chart of the MRR algorithm is shown.

© STM Journals 2021. All Rights Reserved 

| P_ID | P5 | P4 | P3 | P2 | P1 | P2 | P1 | P1 |
|---|---|---|---|---|---|---|---|---|

0 35 78 133 203 273 288 313 323

***AQRR Algorithm***

Here the Gantt chart of the AQRR algorithm is shown.

| P_ID | P5 | P4 | P3 | P2 | P1 | P2 | P1 |
|---|---|---|---|---|---|---|---|

0 35 78 133 197 261 282 323

Average Turnaround time = (323+282+133+78+35)/5 = 170.2
Average Waiting Time = (218+197+78+35+0)/5 = 105.6

**Table 5.** Comparative study of different algorithms with zero arrival time and burst time in descending order.

| Algorithm | Avg. waiting time | Avg. turnaround time |
|---|---|---|
| Round-Robin | 209.4 | 274 |
| SARR | 185.8 | 250.4 |
| DQRRR | 144.8 | 209.4 |
| MRR | 106.8 | 171.4 |
| AQRR | 105.6 | 170.2 |

***In Random Order***

In the following subsection processes with zero arrival time and the same priorities with random burst time are taken for comparison.

**Table 6.** Processes with zero arrival time and burst time in random order.

| Processes | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 105 | 1 |
| P2 | 0 | 60 | 1 |
| P3 | 0 | 120 | 1 |
| P4 | 0 | 48 | 1 |
| P5 | 0 | 75 | 1 |

***Gantt Charts of Algorithms are given below***

***RR Algorithm***

Here the Gantt chart of the RR algorithm is shown.

| P_ID | P1 | P2 | P3 | P4 | P5 | P1 | P2 | P3 | P4 | P5 | P1 | P2 | P3 | P5 | P1 | P3 | P1 | P3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0 25 50 75 100 125 150 175 200 223 248 273 283 308 333 358 383 388 408

***SARR Algorithm***

Here the Gantt chart of the SARR algorithm is shown.

| P_ID | P1 | P2 | P3 | P4 | P5 |
|---|---|---|---|---|---|

0 105 165 285 333 408

***DQRRR Algorithm***

Here the Gantt chart of the DQRRR algorithm is shown.

| P_ID | P4 | P3 | P2 | P1 | P5 | P3 | P1 | P3 |
|---|---|---|---|---|---|---|---|---|

0 48 123 183 258 333 370 400 408

***MRR Algorithm***

Here the Gantt chart of the MRR algorithm is shown.

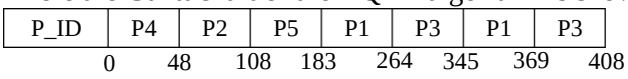
| P_ID | P4 | P2 | P5 | P1 | P3 | P5 | P1 | P3 | P3 |
|---|---|---|---|---|---|---|---|---|---|

0 48 108 180 252 324 327 360 405 408

© STM Journals 2021. All Rights Reserved 

***AQRR Algorithm***

Here the Gantt chart of the AQRR algorithm is shown.




Average Turnaround time = (369+108+408+48+183)/5 = 223.2
Average Waiting Time = (264+48+288+0+108)/5 = 141.6

**Table 7.** Comparative study of different algorithms with zero arrival time and burst time in random order.

| Algorithm | Avg. waiting time | Avg. turnaround time |
|---|---|---|
| Round-Robin | 245.4 | 327 |
| SARR | 177.6 | 259.2 |
| DQRRR | 182.8 | 274.4 |
| MRR | 168.6 | 250.2 |
| AQRR | 141.6 | 223.2 |

***Comparison***

The above algorithms are compared based on five processes with zero arrival time. Three test scenarios are taken: processes with ascending/descending/random order of their burst times.

In Figure 4, the average performance of five algorithms is compared based on the average waiting time and average turnaround time of these algorithms for the above three cases.

On comparing the execution of the proposed algorithm with the other four Round Robin variants, it is clear that the proposed algorithm outshines the other four algorithms in both averages waiting time and average turnaround time. The proposed algorithm has a 43.1% decrease in average waiting time and a 32.3% decrease in average turnaround time compared with classical RR.

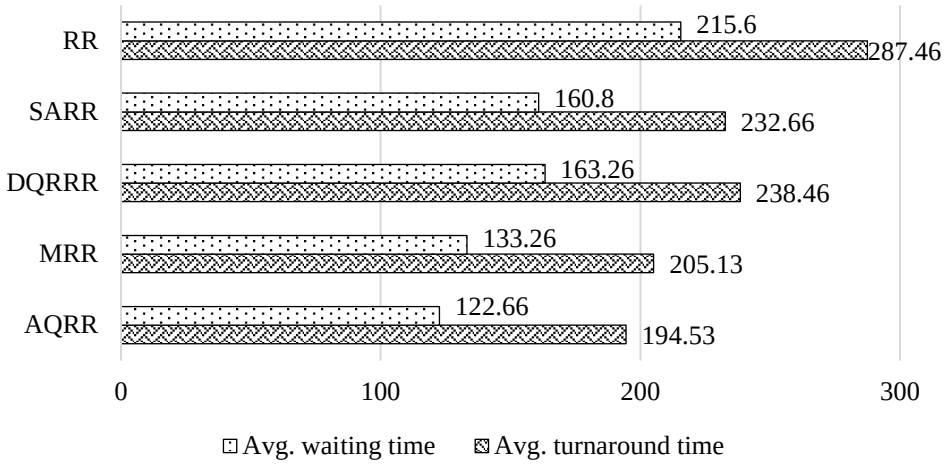



**Figure 4.** Comparative study of avg. waiting time and avg turnaround time among RR, SARR, DQRRR, MRR, and AQRR scheduling algorithms.

**Without Zero Arrival Time, Same Priority**

***In Ascending Order***

In the following subsection processes without zero arrival time and the same priorities with increasing burst time are taken for comparison.

© STM Journals 2021. All Rights Reserved 

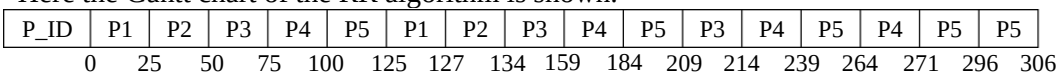
**Table 8.** Processes without zero arrival time and burst time in ascending order.

| Processes | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 27 | 1 |
| P2 | 3 | 32 | 1 |
| P3 | 5 | 55 | 1 |
| P4 | 7 | 82 | 1 |
| P5 | 9 | 110 | 1 |

***Gantt Charts of Algorithms are given below***

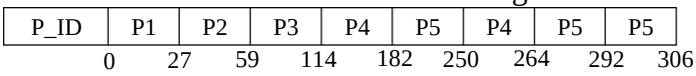
***RR Algorithm***

Here the Gantt chart of the RR algorithm is shown.



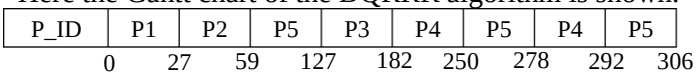

***SARR Algorithm***

Here the Gantt chart of the SARR algorithm is shown.



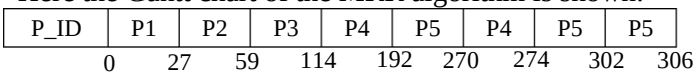

***DQRRR Algorithm***

Here the Gantt chart of the DQRRR algorithm is shown.



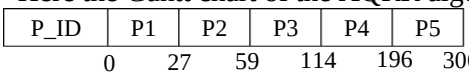

***MRR Algorithm***

Here the Gantt chart of the MRR algorithm is shown.




***AQRR Algorithm***

Here the Gantt chart of the AQRR algorithm is shown.




Average turnaround time = (27+56+109+189+297)/5 = 135.6
Average waiting time = (0+24+54+107+187)/5 = 74.4

**Table 9.** Comparative study of different algorithms without zero arrival time and burst time in ascending order.

| Algorithm | Avg. waiting time | Avg. turnaround time |
|---|---|---|
| Round-Robin | 144.8 | 205.6 |
| SARR | 88 | 149.2 |
| DQRRR | 107.2 | 168.4 |
| MRR | 90 | 151.2 |
| AQRR | 74.4 | 135.6 |

***In Descending Order***

In the following subsection processes without zero arrival time and the same priorities with decreasing burst time are taken for comparison.

***Gantt Charts of Algorithms are given below***

***RR Algorithm***

Here the Gantt chart of the RR algorithm is shown.

© STM Journals 2021. All Rights Reserved 

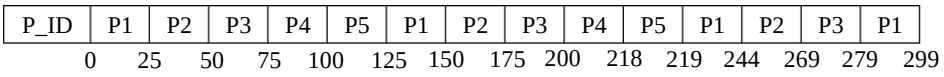
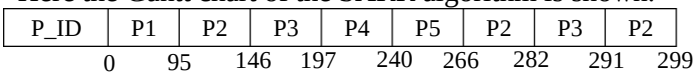



**Table 10.** Processes without zero arrival time and burst time in descending order.

| Processes | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 95 | 1 |
| P2 | 2 | 75 | 1 |
| P3 | 4 | 60 | 1 |
| P4 | 8 | 43 | 1 |
| P5 | 16 | 26 | 1 |

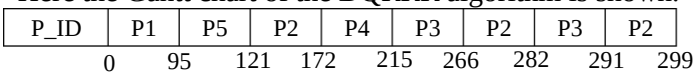
***SARR Algorithm***

Here the Gantt chart of the SARR algorithm is shown.



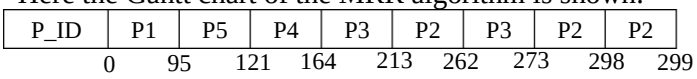

***DQRRR Algorithm***

Here the Gantt chart of the DQRRR algorithm is shown.



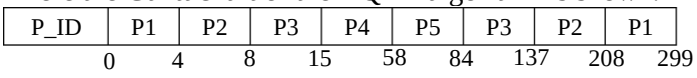

***MRR Algorithm***

Here the Gantt chart of the MRR algorithm is shown.




***AQRR Algorithm***

Here the Gantt chart of the AQRR algorithm is shown.




Average Turnaround time = (299+206+133+50+68)/5 = 151.2
Average Waiting Time = (204+131+73+7+42)/5 = 91.4

**Table 11.** Comparative study of different algorithms without zero arrival time and burst time in descending order.

| Algorithm | Avg. waiting time | Avg. turnaround time |
|---|---|---|
| Round-Robin | 191 | 250.8 |
| SARR | 172.4 | 232.2 |
| DQRRR | 138.4 | 198.2 |
| MRR | 124.6 | 184.4 |
| AQRR | 91.4 | 151.2 |

***In Random Order***

In the following subsection processes without zero arrival time and the same priorities with random burst time are taken for comparison.

**Table 12.** Processes without zero arrival time and burst time in random order.

| Processes | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 45 | 1 |
| P2 | 5 | 90 | 1 |
| P3 | 8 | 70 | 1 |
| P4 | 15 | 38 | 1 |
| P5 | 20 | 55 | 1 |

© STM Journals 2021. All Rights Reserved 

***Gantt Charts of Algorithms are given below***

***RR Algorithm***

Here the Gantt chart of the RR algorithm is shown.

| P_ID | P1 | P2 | P3 | P4 | P5 | P1 | P2 | P3 | P4 | P5 | P2 | P3 | P5 | P2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0 25 50 75 100 125 145 170 195 208 233 258 278 283 298

***SARR Algorithm***

Here the Gantt chart of the SARR algorithm is shown.

| P_ID | P1 | P2 | P3 | P4 | P5 | P2 | P3 | P5 | P2 |
|---|---|---|---|---|---|---|---|---|---|

0 45 99 153 191 245 261 277 278 298

***DQRRR Algorithm***

Here the Gantt chart of the DQRRR algorithm is shown.

| P_ID | P1 | P4 | P2 | P5 | P3 | P2 | P3 | P2 |
|---|---|---|---|---|---|---|---|---|

0 45 83 145 200 262 280 288 297

***MRR Algorithm***

Here the Gantt chart of the MRR algorithm is shown.

| P_ID | P1 | P4 | P5 | P3 | P2 | P5 | P3 | P2 | P2 |
|---|---|---|---|---|---|---|---|---|---|

0 45 83 135 187 239 242 260 295 298

***AQRR Algorithm***

Here the Gantt chart of the AQRR algorithm is shown.

| P_ID | P1 | P2 | P3 | P4 | P5 | P3 | P2 |
|---|---|---|---|---|---|---|---|

0 45 75 80 118 173 238 298

Average Turnaround time = (45+293+230+103+153)/5 = 164
Average Waiting Time = (0+203+160+65+98)/5 = 105

**Table 13.** Comparative study of different algorithms without zero arrival time and burst time in random order.

| Algorithm | Avg. waiting time | Avg. turnaround time |
|---|---|---|
| Round-Robin | 173.2 | 232.8 |
| SARR | 148.6 | 208.2 |
| DQRRR | 113.6 | 173.2 |
| MRR | 116.4 | 176 |
| AQRR | 105 | 164 |

***Comparison***

The above algorithms are compared based on five processes with different arrival times. Three test scenarios are taken—processes with ascending/descending/random order of their burst times.

In Figure 5, the average performance of five algorithms is compared based on the average waiting time and average turnaround time of these algorithms for the above three cases.

With a 46.8% decrease in average waiting time and 34.6% decrease in average turnaround time with comparison to Classical RR, the proposed algorithm performs best among all the five variants of RR taken for comparison.

**With Different Priority**

In the following subsection processes, different priorities with random burst time are taken for comparison (Figure 6).

© STM Journals 2021. All Rights Reserved 

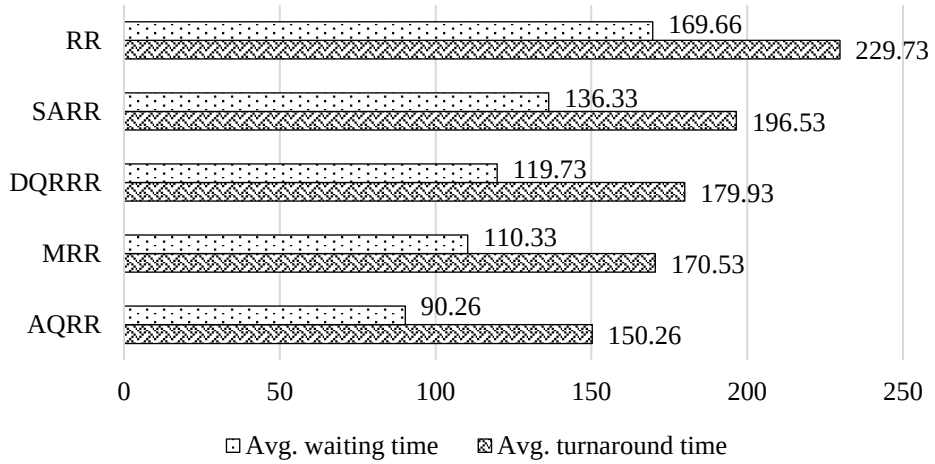



**Figure 5.** Comparative study of avg waiting time and avg turnaround time between RR, SARR algorithms.

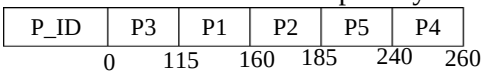
**Table 14.** Processes with zero arrival time and burst time in random order and different priority.

| Processes | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 45 | 4 |
| P2 | 0 | 25 | 4 |
| P3 | 0 | 115 | 5 |
| P4 | 0 | 20 | 3 |
| P5 | 0 | 55 | 3 |

***Gantt Charts of Algorithms are given below***

***Priority Based Algorithm***

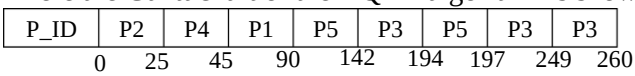
Here the Gantt chart of the priority-based algorithm is shown.




Average turnaround time = (160+185+115+260+240)/5 = 192
Average waiting time = (115+160+0+240+185)/5 = 140

***AQRR Algorithm***

Here the Gantt chart of the AQRR algorithm is shown.




Average Turnaround time = (90+25+260+45+197)/5 = 123.4
Average Waiting Time = (45+0+145+25+142)/5 = 71.4

**Table 15.** Comparative study between processes with zero arrival time and burst time in random order and different priority.

| Algorithm | Avg. waiting time | Avg. turnaround time |
|---|---|---|
| Priority-based | 140 | 192 |
| AQRR | 71.4 | 123.4 |

## RESULT ANALYSIS

From the above analysis, it is clear that the proposed algorithm performs better than all the mentioned algorithms in all test cases.

© STM Journals 2021. All Rights Reserved 

Figure 7 gives the percentage of waiting time saved by each algorithm concerning traditional RR. The Proposed AQRR performs 44.71% better than traditional RR in this field.

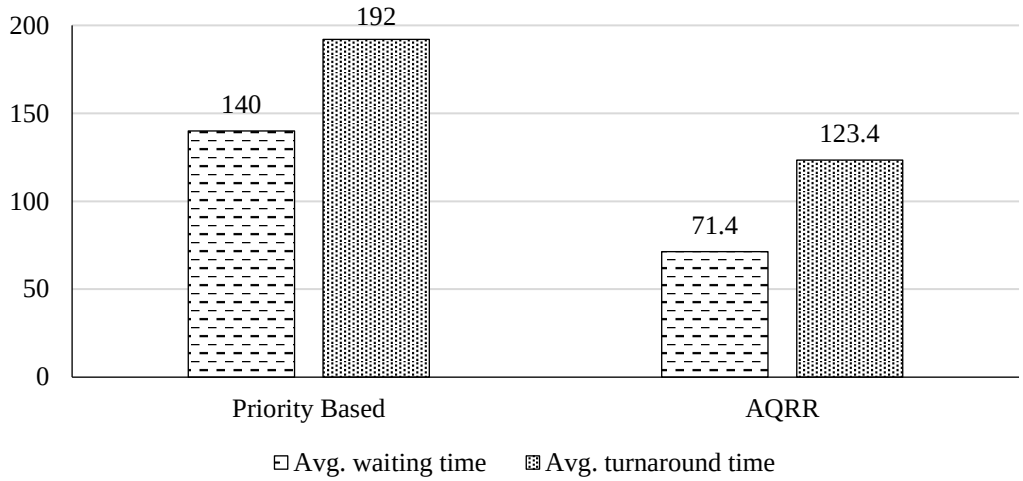



**Figure 6.** Graphical representation of comparison between preemptive priority based and AQRR scheduling.

**Table 16.** Percentage of waiting time reduced by each algorithm.

| | **Waiting time** | | | | |
|---|---|---|---|---|---|
| | ***RR*** | ***SARR*** | ***DQRRR*** | ***MRR*** | ***AQRR*** |
| **Grand total** | 1155.40 | 891.40 | 859.00 | 730.80 | 638.8 |
| **Percentage gain** | 0.00% | 22.85% | 25.65% | 36.75% | 44.71% |

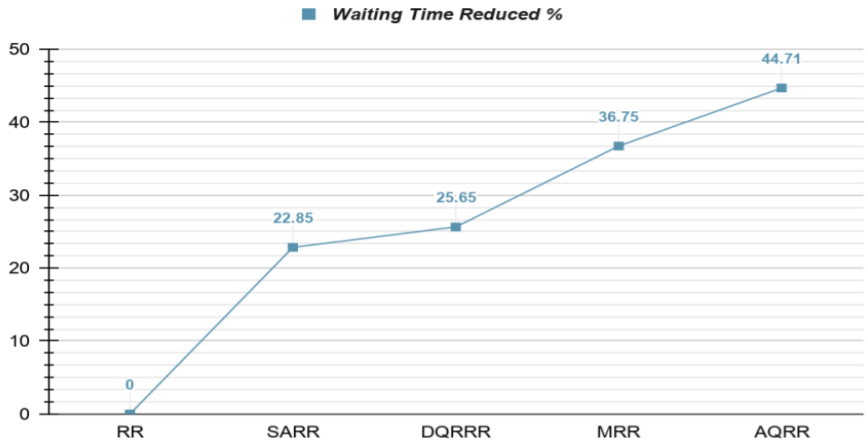



**Figure 7.** Percentage of waiting time reduced.

**Table 17.** Percentage of turnaround time reduced by each algorithm.

| | **Turnaround time** | | | | |
|---|---|---|---|---|---|
| | ***RR*** | ***SARR*** | ***DQRRR*** | ***MRR*** | ***AQRR*** |
| Grand total | 1551.60 | 1287.60 | 1255.20 | 1127.00 | 1034.40 |
| Percentage gain | 0.00% | 20.1% | 19.1% | 27.37% | 33.33% |

Figure 8 gives the percentage of turnaround time saved by each algorithm concerning traditional RR. The Proposed AQRR performs 33.33% better than traditional RR in this field.

Table 15 and 16 Grand Total is calculated by summing up the respective values obtained for these algorithms from Tables 2, 4, 6, 8, 10, 12.

© STM Journals 2021. All Rights Reserved 

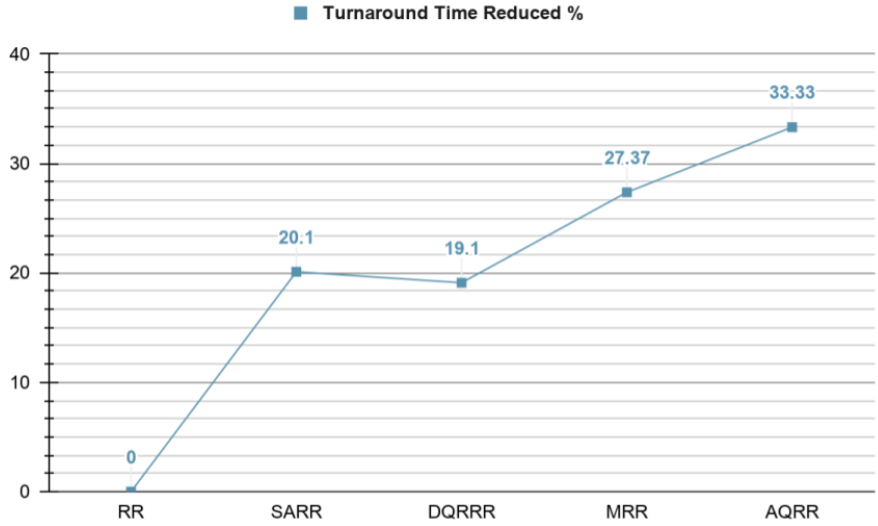



**Figure 8.** Percentage of turnaround time reduced.

## CONCLUSIONS

The paper presents a new variant of scheduling algorithms based on the standard Round Robin algorithm but incorporated with custom dynamic quantum in a uniprocessor environment. The proposed algorithm allows not only faster and efficient execution of the processes but also reduces starvation among processes. The result comprises the comparative study of the average waiting time and average turnaround time using different scheduling algorithms for various test cases. The AQRR scheduling algorithm imparts good performance with respect to average waiting time and average turnaround time. In comparison to other existing 4 algorithms the percentage gain in avg. waiting time and avg. turnaround time is much higher as shown. With a 44.71% gain on avg. waiting time and 33.33% over the traditional Round-Robin algorithm proves to be both efficient and optimized.

The future of this work involves the enrichment of this algorithm in a multiprocessing environment.

## Acknowledgments

We wish to thank many people for their direct or indirect help in achieving this goal. Special thanks to our parents for their support and encouragement throughout our study, their faith in us ignited that spark to do anything for their happiness.

## REFERENCES

1. Hanish AA. Operating systems and communication protocols. Proceedings of the international workshop on object orientation in operating systems; 1995. p. 166-70. doi: 10.1109/IWOOS.1995.470561.
2. Process scheduling: long, medium, short-term scheduler. Available from: guru99.com [cited Feb 1, 2021]. Available from: https://www.guru99.com/process-scheduling.html.
3. Baruah Sanjoy. The limited-preemption uniprocessor scheduling of sporadic task systems 17th Euromicro Conference on Real-Time Systems (ECRTS'05); 2005. p. 137-44. doi: 10.1109/ECRTS.2005.32.
4. Kameda H. CPU scheduling for effective multiprogramming. Lecture Notes in Computer Science IBM Germany Sci Symposium Series 1980 Oct 1. 1982:(104-18). doi: 10.1007/3-540-11604-4_50.
5. Goel Neetu, Garg RB. A comparative study of CPU scheduling algorithms. Int J Graph Image Process. November 2012, Available from: arXiv:1307.4165 [cs.OS];2.
6. Mili. Patel, Rakesh P, SJRR CPU scheduling algorithm International Journal of Engineering and Computer Science. 2013;2(12).

© STM Journals 2021. All Rights Reserved 

7. Dash Amar Ranjan, Sahu Sk, Samantra SK. An optimized round robin CPU scheduling algorithm with dynamic time quantum. IJCSEIT. 2015;5(1, February):07-26. doi: 10.5121/ijcseit.2015.5102.
8. Fang Z. A weight-based multiobjective genetic algorithm for Flowshop scheduling International Conference on Artificial Intelligence and Computational Intelligence. Vol. 2009; 2009. p. 373-7. doi: 10.1109/AICI.2009.130.
9. Arunekumar NB, Kumar A, Joseph KS. Hybrid bat inspired algorithm for multiprocessor real-time scheduling preparation International Conference on Communication and Signal Processing (ICCSP). Vol. 2016; 2016. p. 2194-8. doi: 10.1109/ICCSP.2016.7754572.
10. Patel Jyotirmay, Solanki AK. Performance evaluation of CPU scheduling by using hybrid approach. Int J Eng Res Technol (IJERT). 2012;1(June).
11. Perry Marcus B. The Weighted Moving Average Technique. Wiley encyclopedia of operations research and management science; February 15, 2011. doi: 10.1002/9780470400531.eorms0964.
12. Balharith T, Alhaidari F. Round Robin Scheduling Algorithm in CPU and Cloud Computing: a review 2nd International Conference on Computer Applications & Information Security (ICCAIS). Vol. 2019; 2019. p. 1-7. doi: 10.1109/CAIS.2019.8769534.
13. Rajguru AA, Apte SS. A performance analysis of task scheduling algorithms using qualitative parameters. Int J Comput Appl;74(19):33-8. doi: 10.5120/13004-0308.
14. Dhotre S, Patil S. Cause of process starvation for Linux completely fair scheduler with Apache server International Conference on Computing Methodologies and Communication (ICCMC). Vol. 2017; 2017. p. 814-9. doi: 10.1109/ICCMC.2017.8282579.
15. Matarneh Rami J. Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes. Am J Appl Sci. 2009;6(10):1831-7. doi: 10.3844/ajassp.2009.1831.1837.
16. Dr. Behera HS, Mohanty R, Nayak D. A new proposed dynamic quantum with Re-adjusted round robin scheduling algorithm and its performance analysis. Int J Comput Appl. August 2010;5(5):10-5. doi: 10.5120/913-1291.
17. Singh M, Agrawal R. Modified Round Robin algorithm (MRR) IEEE International Conference on Power, Control, Signals and Instrumentation Engineering. 2017; p. 2832-9. doi: 10.1109/ICPCSI.2017.8392238.

© STM Journals 2021. All Rights Reserved 