An Improved Round Robin Algorithm for an Efficient CPU Scheduling

A.M.Oshani Bandara
Faculty of Graduate Studies and Research
Sri Lanka Institute of Information Technology
Malabe, Sri Lanaka
poornibandara14@gmail.com

U.U.Samantha Rajapaksha
Department of Information Technology
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
samantha.r@sliit.lk

2022 2nd International Conference on Advanced Research in Computing (ICARC) | 978-1-6654-0741-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICARC54489.2022.9754037

***Abstract*—The algorithm policy applied by a CPU, to schedule running processes, has an impact on the efficiency of an operating system (OS). As a result, a superior CPU scheduling algorithm leads to higher OS performance while using limited resources and for shorter periods of time. As a result, several strategies have been suggested and implemented to improve CPU scheduling performance. Any scheduler's primary responsibility is to assign jobs to the most effective and reliable resource available. It's also a fact that if jobs and resources aren't planned properly, the entire operating system's productivity suffers significantly. The "Round Robin" is deemed an effective and fair approach because of each process is allocated the same amount of time quantum. Nevertheless, the effectiveness of the system is determined by the selected time-quantum. The major goal of this research is to develop the present round-robin algorithm by enhancing the time quantum for candidate processes in real-time in such a way that its impartiality is maintained. There is no loss of property. This paper's proposed algorithm finds the time left in a process's last turn, and then Determines whether it is time-based on a certain threshold value, should quantum be expanded or not. An arithmetic simulation has been created to demonstrate that the suggested method is correct. It outperforms the traditional round-robin algorithm. In words of several performing metrics such as average waiting time, context switches and the number of turnaround times, the suggested adjusted version of the round-robin algorithm outperforms the traditional round-robin algorithm, according to the results of the experimental investigation.**

***Index Terms*—CPU Scheduling, Round-Robin Algorithm, Time quantum, Turnaround time, Waiting time, Response Time, Context Switching.**

I. Introduction

Scheduling is an elementary function in "operating system", because virtually the entire computer resources are scheduled before utilizing. It entails allocating resources and time to processes or tasks in order to achieve specific performance requirements. A computer with multi-programming has different processes challenging for the CPU at the same time. Consequently, a decision necessary to be undertaken about which process to run next, which is handled by the scheduler [1].

A scheduler uses scheduling algorithms to carry out the process. As a result, the CPU scheduling algorithms are at the operating system's core [1]. CPU scheduling is a built-in feature of operating systems. When there are several processes in the ready queue to execute, the operating system uses a scheduler (which is using a scheduling algorithm) to determine which process will be executed first. "FCFS (First Come, First Serve), RR (Round Robin), SJF (Shortest Job First), and Priority Scheduling" are some of the scheduling algorithms available. The goal of these scheduling algorithms is to decrease "turnaround-time, response-time, waiting-time, and the number of context switches." There are several scheduling criteria and Thus analyze and decide which scheduling algorithm is the best based on these criteria [1]. To distribute the CPU to the procedures waiting in the prepared line, a CPU scheduling algorithm is used.

Multiple processes; are stored and maintained in the main memory in multi-programming systems. The insides of the "Central Processing Unit" (CPU) registers used by the process, as well as the slice of memory that includes the program that is being executed by the process (along with its related data), define each process. In addition, each process alternates among processor use and the occurrence of "input/output" (I/O) events. The latter may be a waiting period for an Input/Output or some other external occurrence to happen. One operation will be executed at a time by the processor, then if the latter is waiting for an event, it will be switched to another.

A CPU-scheduler is a component of operating systems which manages privileges to the processor. The scheduler is primarily concerned through "turnaround time, response time, waiting time, fairness" etc. Scheduling is the key to multi-programming. one of the most critical characteristics that impact the effectiveness is CPU scheduling [2].

Multi-level queue and multi-level feedback CPU scheduling techniques can be used to schedule a CPU that has several types of processes that must run. The importance of multilevel queue scheduling is well known, and it is widely used to schedule jobs in a processor [2].

CPU scheduling algorithms select the next process for execution. Context switches often happen during CPU scheduling. When a context switch occurs, the currently running process is terminated and CPU time is given to another process. The CPU's performance reduces as a result. An effectual scheduling algorithm is required to maximize CPU usage. Various CPU scheduling algorithms have various features, and one system metric may be preferred over another [3].

978-1-6654-0741-0/22/$31.00 ©2022 IEEE

When deciding which algorithm is best in a given condition, must deliberate the algorithm's various performance characteristics and properties. When choosing a process to run, CPU efficiency is a critical factor to consider. It is never an idle to leave the CPU unattended. As a result, CPU scheduling must be completed in such a manner that the system is used systematically and efficiently [3].

Round Robin is a broadly used scheduling algorithm that gives every single process equal priority. Any phase in system is blocked after a predetermined time quantum or time slice. RR, on the other hand, improves response time and makes optimal use of shared resources [4]. The drawbacks of RR include undesirable overhead, longer waiting times, longer turnaround times for processes with variable CPU bursts due to the use of static time quantum, among other things. In this case, a RR on the sorted ready queue with progressive time quantum can be created. For the execution of the operation, Round Robin uses a slight unit of time known as Time Slice or Time Quantum. If the CPU burst of a process reaches 1-time quantum, the process is pre-empted and returned to the ready queue [4].

This research paper includes following sub sections. section two described the different enhanced solutions for the round robin algorithm using modified time quantum by researches. section three described the proposed algorithm and result and evaluation discussion included in the fourth section. And finally conclusion and future works explained in the latter subsections of this paper.

II. Literature Review

Since the time quantum selection is such a crucial factor in the RR algorithm's effectiveness, numerous researchers have attempted to enhance it by suggesting alternative approaches for calculating the ideal time quantum .Several attempts have been made in recent years to develop the classic Round Robin algorithm. All those efforts used conventional methods in addition focused on the algorithm's central component, the time quantum (TQ). Static TQ was used in some of these algorithms, while dynamic TQ was used in others. Furthermore, a few efforts remained, completed to increase the effectiveness of the RR algorithm by predicting the proper value of TQ using machine learning approaches [5] [6].

This section delivers a summarized overview of the "Improved RR Approaches" initiate in the background studies.

The RR-algorithm had better be improved, according to the S. Arif et al., [7]. Their approach is efficacious, "if the remaining time of the process under execution is less than one Time Quantum." The technical writers of [8] proposed a innovative modification for the RR algorithm that is similar to the prior one except that the processor will be allocated again if the currently running process's outstanding CPU time is not equivalent to or greater than one Time Quantum.

In her paper, J. Khatri [9], in 2016 suggested a slight improvement on the algorithm proposed by [5]. Her enhancement involves determining if the process's remaining CPU time is small than one Time Quantum.

In another survey, S. Hiranwal et al. [5], the authors suggested an upgraded method to improve Round Robin algorithm. Their system calculates the Time Quantum based on the no of processes. If the number is even, the Time Quantum equals the means of every process burst times; or else, the Time Quantum equals the CPU time of the middle process.

S.K. Panda and S.K. Bhoi [10] introduced an enhanced method for calculating Time Quantum dynamically at each step built on the variance among the minimum and maximum CPU time values for the processes. If calculated Time Quantum is less than 25, the TQ is set to 25, otherwise the current calculated TQ is used.

Matarneh [11] suggested a new technique called Self-Adjustment-Round-Robin (SARR). Time Quantum is dynamically changed on the base of this. Every iteration, the value of time quantum is dynamically allocated depended on the current process's burst time. Alternative CPU scheduling algorithm is proposed in [12]. The minimum and maximum CPU burst times are determined here, and TQ is changed by increasing the combination of maximum and minimum CPU burst time.

An enhanced form of the Round Robin Algorithm was proposed by P. Banerjee et al. in 2012 [13]. To change the value of TQ, a new criterion is added. Varma et al. [14] introduced another flavour of RR in the same year. The value of TQ is calculated using the square root equation.

In 2014, Mishra [15] projected a new algorithm that uses the features of SJF and RR to set the value of TQ. Another algorithm was proposed by Nandal and Shyam [16]. They calculated the highest and mean burst time values to determine TQ. [17] proposes a new RR algorithm based on a group of processes and the values of the maximum and minimum CPU burst times. The majority of current solutions result in longer waiting times due to complicated mechanisms that take into observation the complex TQ calculation.

In this paper, I review the literature for various enhanced Round Robin-based scheduling algorithms and suggest a novel method for improving the performance of RR-based algorithms. The proposed method has undergone extensive testing to determine its effectiveness.

Round robin is a commonly used scheduling algorithm that assigns equal priority to all processes. After a predetermined time, "quantum or time slice", any phase in the system is preempted. RR, on the other hand, improves response time and makes optimal use of shared resources. The drawbacks of RR include undesirable overhead, longer waiting times, longer turnaround times for processes with variable CPU bursts due to the use of static time quantum, among other things. In this case, an RR on the sorted ready queue with progressive time quantum can be created. Round Robin uses a minor unit of time known as "Time Slice" or "Time Quantum" to carry out the operation.

The major contribution of this research is enhancing the round-robin algorithm by proposing a novel technique named IRR ("Improved Round Robin"). It concentrates on providing a solution for the time quantum problem by calculating the

mean for all the tasks in the ready queue, which is sorted based on the SJF manner.

The process of tuning the time quantum dynamically is repeated for each task separately and for each round. Moreover, checking the remaining burst time of the task is an essential principle applied with proposed algorithm. If the remaining burst time is less than or equal to the current task quantum, the task execution is completed and then removed from the ready queue.

III. Proposed Solution

The suggested IRR technique was built and tested in a java simulation framework, which is extensible and fully featured. It is built on the framework and enables for the modeling, simulation, and testing of computing infrastructure and application services. The ability to combine modeling and analyze application services was the primary reason for selecting this simulation tool. It facilitates task scheduling strategies, network connection configuration, data center resource energy modeling, and the provisioning of different workloads.

Algorithm 1 the Pseudocode of the RR Algorithm in CPU Scheduling

Declarations
RQ : Ready Queue
TQ : Time Quantum
T_i : task i
t : Timer
WHILE (RQ != NULL)
BEGIN
//Keep the ready queue as a FIFO queue of tasks
IF (a new task is arrived)
BEGIN
//Set a timer to interrupt after one time slot and dispatch the tasks.
BEGIN
// task may have executed less than one **TQ**
CASE : T_i release the resources voluntarily
CASE: Scheduler proceed nest task in **RQ**
IF (Running task is > TQ)
THEN
// Timer will go off
An interruption to the OS
END
END
END IF
END

As can be seen from the preceding explanation, the efficiency of the RR algorithm is determined by the TQ size. As a result, selecting the TQ size is a significant problem for increasing the RR algorithm's overall performance. If the TQ is too long, RR tends to become an FCFS algorithm, whereas if the TQ is too slight, RR may perform badly due to the high expense of context transitions. The pseudo-code of the RR method as described in is shown in Algorithm 1.

Figure 1 illustrates the specifics of the primary phases involved in the proposed solution. In addition, this flowchart illustrating the proposed approach's technique.

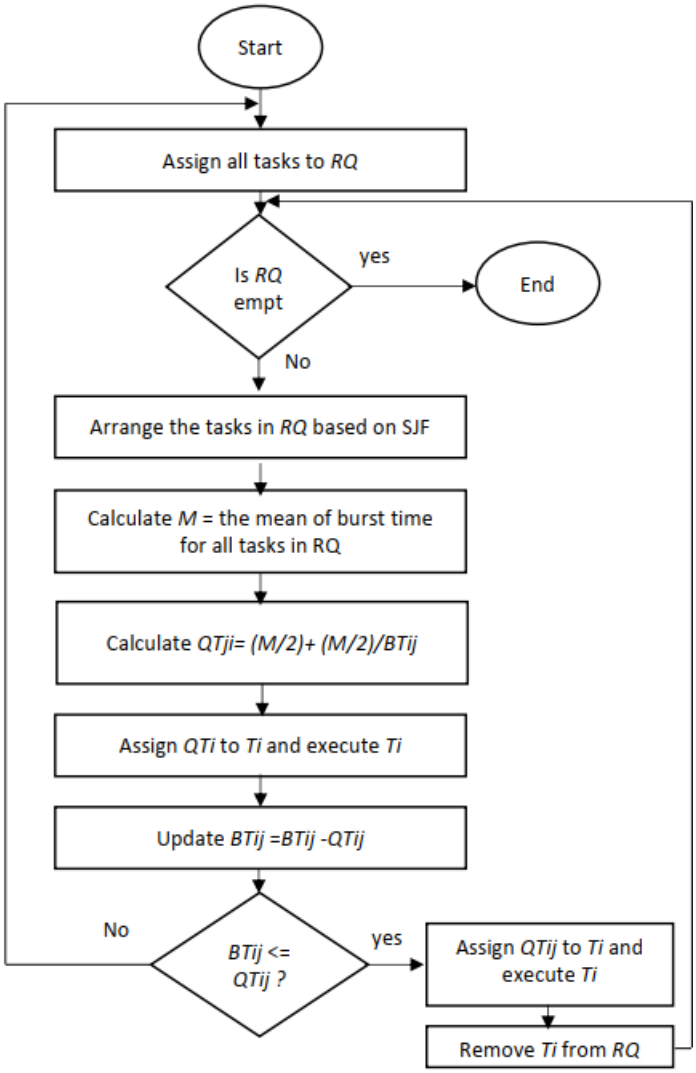

Fig. 1. The proposed algorithm's flow chart.

The primary impact of this section is to suggest a novel method concentrating on the usual RR algorithm drawbacks. The proposed prototype enhances the functionality of the classic RR algorithm for task scheduling by reducing average turnaround time, average waiting time, and average response time by optimizing performance metrics.

In this recommended IRR approach, tasks are initially gathered from users and placed in the ready queue in the order of their arrival (FCFS). The ready queue receives each job, which is subsequently sorted using the SJF algorithm. The average mean of all tasks in the ready queue is calculated each round. "Equation one - (1)" is then used to calculate the time quantum for each task. When the remaining burst time is less

than or equal to the job's quantum, the task is finished and removed from the ready queue. If the job is not completed in the current round, it will be saved to the end of the ready queue and executed in the next.

Furthermore, New jobs are added to the ready queue and stored there until the current round is completed:

$$TQij = (m/2) + (m/2)/BTij \quad (1)$$

TQij = time quantum - task i in round j

m = average mean at the ready queue

BTij = burst time - task i in round j

The IRR is arranged of 07 main stages, and they are as bellow:

Step 1: Based on their burst time, arrange the submitted tasks in ascending order.

Step 2: For all jobs in the ready queue, compute their arithmetic mean.

Step 3: Estimate the amount of the time quantum based on "Equation (1)".

Step 4: Implement all tasks based on their computed time quantum.

Step 5: When a novel task enters, do the following:

1. Sort-out all tasks in the ready queue in the SJF (Shortest Job First).

2. All of the tasks in the ready queue will have their m and TQij values updated.

Step 6: The m and TQij values for all the tasks in the ready queue will be adjusted after a task is completed.

Step 7: Until all the tasks are completed, repeat steps.

IV. Evaluation and Discussion

The suggested IRR technique was assessed using the resulting four performance metrics:

The overall amount of time spent, or the amount of time spent waiting; Response-time, which was the time it took from the task's arrival to the commencement of its implementation; Turnaround time, was the amount of time that passed between the task's arrival and completion; and Context switches or how many times the task status moved from one action to the next.

To examine the suggested model, Thus looked at two different scenarios:

(1) Estimating the suggested model by comparison it to the FCFS and SJF algorithms in order to determine the impact of combining the two algorithms.

(2) Using the existing solutions to compare the proposed model to comparable methods, and evaluating the proposed model's performance using a real data-set.

in this experiment, the proposed model was validated by conducting several experiments using the NASA data-set. [21] [22]

NASA [21] [22] provided the data-set for this experiment, which has been utilized in other studies such as [23] - [25]. Only the execution time was considered in this experiment. This is due to the fact that the tasks in the NASA data-set came one by one, with no overlap in their arrival timings. As a result, these arrival times were not adequate for validating the suggested model.

TABLE I
Data Sets with Zero Arrival Time. Case 1 represents Increasing order. Case 2 represents Decreasing Order and Case 3 represent Random order for the Arrival time and Burst time

Case 1 Arrival Time	Case 1 Burst Time	Case 2 Arrival Time	Case 2 Burst Time	Case 3 Arrival Time	Case 3 Burst Time
0	30	0	77	0	80
0	34	0	54	0	45
0	62	0	45	0	62
0	74	0	19	0	34
0	88	0	14	0	78

TABLE II
Data Sets with non - Zero Arrival Time. Case 1 represents Increasing order. Case 2 represents Decreasing Order and Case 3 represent Random order for the Arrival time and Burst time

Case 1 Arrival Time	Case 1 Burst Time	Case 2 Arrival Time	Case 2 Burst Time	Case 3 Arrival Time	Case 3 Burst Time
0	14	0	80	0	65
2	34	2	74	1	72
6	45	3	70	4	50
8	62	4	18	6	43
14	77	5	14	7	80

A. Proportional Study on the Suggested Model (IRR) with SJF and FCFS

As indicated in Tables 1 and 2, the data set contained of 02 experiments, the first of which expected zero arrival times and the second of which took different arrival times into account. Furthermore, based on the order of the burst duration, each experiment had three episodes (increasing, decreasing, and random). In all three instances, including the SJF algorithm into the suggested model resulted in significant reductions in average turnaround time, average waiting time, and response time, as shown in Figure 2.

It also kept track of how many context switches are made. As a result, Thus discovered that combining the SJF algorithm with the suggested model improved overall performance in a surprising way when compared to combining the FCFS method with the proposed model.

B. Proportional Study on the Planned Model (IRR) and the Interrelated Algorithms

Several tests were conducted to assess the suggested model's performance. The benchmark was taken from many methods and simulated with the same settings, taking into account the task arrival time and burst time, for a fair evaluation.

As follows, Thus conducted four distinct assessments, comparing suggested algorithm to several comparable research. (1) The first test was an assessment of work in comparison to that reported in [18]. (2) The second test was an assessment of work in comparison to that published in [19], [20].

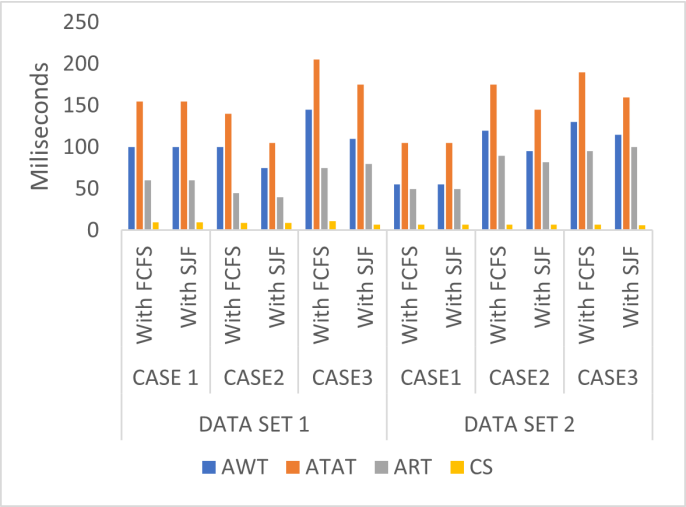

Fig. 2. Comparison of the IRR model with SJF and FCFS.

The data-set for this test was collected from [18] and consisted of three cases. The average turnaround time (ATAT), average waiting time (AWT), and the no of context switches (CS) were used to make the comparison. When compared to the bench-marked methods, the proposed model outperformed the bench-marked algorithms in all three scenarios.

The data-set for this experiment was taken from [19], which included three different examples. The proposed model was assessed and compared to the classic RR algorithm as well as the bench-marked approach in [20] in each situation. In addition, the evaluation was based on the average turnaround time, average wait time, and number of context switches.

In contrast, the performance of the suggested method was reported to be very high in all of the examined metrics in other evaluated scenarios.

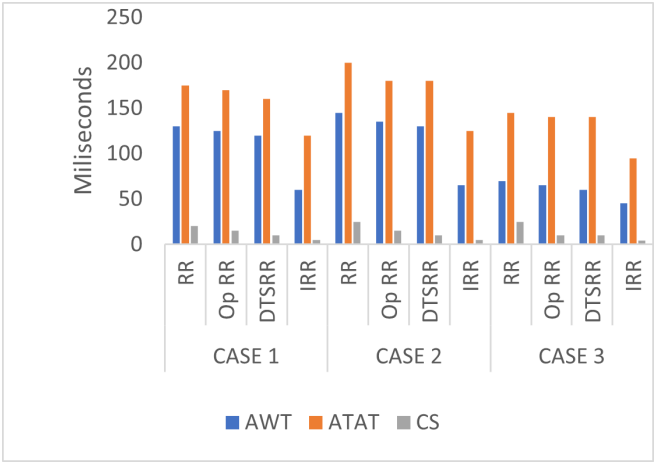

Fig. 3. Comparison of the results with existing solutions.

The data-set for this evaluation was collected from [19], and it was divided into two studies based on the task arrival time. Each one was separated into two cases. By comparing the suggested model to the classic RR and DTSRR algorithms, the data-set was applied to evaluate the suggested model.

TABLE III
COMPARISON OF THE IMPROVEMENTS WITH EXISTING SOLUTIONS.

Metrics	RR	IRR	DTSRR	OP RR
Avg. WT	12%	56.5%	29%	29%
Avg. TAT	6%	38%	20%	13%
No of CS	12%	65%	16%	11%

The suggested model's performance was evaluated using the average waiting time, average turnaround time, and average reaction time from the prior examples in Table 3 under the identical simulation settings.

V. CONCLUSION

The "Time Quantum" is a major character in round robin scheduling. This study proposes an improved version of the round robin scheduling technique. For methods that only require a portion of the time specified in the fixed time quantum, this method extends the time quantum. The mathematical model demonstrates that the suggested method's worst case is equivalent to a typical round robin algorithm's best/worst situation. The proposed method, according to the data, the traditional round robin scheduling algorithm in terms of results while having no influence on response time. The major contribution of this study is a new approach called IRR that may be used to improve the traditional Round-Robin algorithm. It focuses on computing the mean for all jobs/tasks in the ready queue, which is sorted using the SJF approach, in order to discover a solution to the time quantum problem. For each job/task and each round, the operation of dynamically adjusting the time quantum is repeated. Furthermore, in the recommended approach, monitoring the task's remaining burst time is a crucial component. The task is completed and removed from the ready queue if the remaining burst duration is less than or equal to the current job quantum.

VI. LIMITATION OF RESEARCH WORK

The Improved Round Robin (IRR) with Smart Time Quantum performed better than the traditional CPU scheduling method in the previous chapter, according to the results. In this section, discusses the limitations of the research.

1.The proposed algorithms (IRR in Uniprocessor, IRR in Multi-Core Processing System) are tested on a non-real-time operating system.

2. Independent processes have been used to test the proposed method (IRR in Multi-Core Processing System.

VII. FUTURE WORK

This research has been defined as a preliminary step for scientists, because there are still some problems which can be cleared up and enhance in future works, including such (1) enhancing the RR algorithm by trying to find a novel paradigm for time quantum calculation that manages to combine dynamic and fixed quantum values to improve RR algorithm efficiency, and (2) using modern tactics such

as neural networks and fuzzy logic to instantly determine the best possible quantum values of tasks.

REFERENCES

[1] K. Dev and S. Reda, "Scheduling challenges and opportunities in integrated CPU+GPU processors," 2016 14th ACM/IEEE Symposium on Embedded Systems for Real-time Multimedia (ESTIMedia), Pittsburgh, PA, USA, 2016, pp. 1-6.

[2] Md. Mamunur Rashid and Md. Nasim Adhtar; "A New Multilevel CPU Scheduling Algorithm", Journals of Applied Sciences 6 (9): 2036-2039,2009.

[3] S. Huajin', G. Deyuan, Z. Shengbing, W. Danghui;"Design Fast Round Robin Scheduler in FPGA", 0-7803-7547-5/021/17.00 @ 2002 IEEE.

[4] S.Panda, S.Bhoi , "An Effective Round Robin Algorithm using Min-Max Dispersion Measure," International Journal on Computer Science and Engineering, vol. 4, no. 1, pp. 45-53, 2012.

[5] S. Hiranwal, Dr. K.C. Roy, "Adaptive Round Robin scheduling using shortest burst approach based on smart time Slice", International Journal of Computer Science and Communication, Vol 2 (2), pp. 319- 323, (July-December) 2021.

[6] Tawfeek, M.; El-Sisi, A.; Keshk, A.; Torkey, F. Cloud task scheduling based on ant colony optimization. Int. Arab J. Inf. Technol. 2015, 12, 129–137.

[7] S. Arif, S. Rehman and F. Riaz, "Design of a modulus based Round Robin scheduling algorithm," 2015 9th Malaysian Software Engineering Conference (MySEC), Kuala Lumpur, Malaysia, 2015, pp. 230- 235, doi: 10.1109/MySEC.2015.7475226.

[8] S. K. Dwivedi and R. Gupta, "A simulator-based performance analysis of multilevel feedback queue scheduling," 2014 International Conference on Computer and Communication Technology (ICCCT), Allahabad, India, 2014, pp. 341-346, doi: 10.1109/ICCCT.2014.7001516

[9] J. Khatri. "An Improved Dynamic Round Robin CPU Scheduling Algorithm Based on Variant Time Quantum". IOSR Journal of Computer Engineering. vol 18.pp 35-40, 2016. 10.9790/0661-1806043540.

[10] S.K. Panda, S.K. Bhoi, "An effective Round Robin algorithm using Min-Max dispersion measure", International Journal on Computer Science and Engineering, Vol 4(1), pp. 45-53, 2012.

[11] R.Matarneh , "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes," American Journal of Applied Sciences, vol. 6, no. 10, pp. 1831-1837, 2009.

[12] U. Saleem and Dr. M. Y. Javed, "Simulation of CPU Scheduling Algorithm", 0-7803-6355-8/00/ 10.00 @ 2000 IEEE.

[13] P. Banerjee, P. Banerjee, S. Sonali Dhal, "Comparative performance analysis of average max Round Robin scheduling algorithm (AMRR) using dynamic time quantum with Round Robin scheduling algorithm using static time quantum", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol 1(3), pp. 56-62, August 2012.

[14] P.S. Varma, "A finest time quantum for improving shortest remaining burst round robin (srbrr) algorithm". J. Glob. Res. Comput. Sci. 2013, 4, 10–15.

[15] M.K.Mishra, F. Rashid, An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum. Int. J. Comput. Sci. Eng. Appl. 2014, 4, 1–8.

[16] R. Shyam S. Nandal , "Improved Mean Round Robin with Shortest Job First Scheduling," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 7, pp. 170-179, 2014.

[17] T. Balharith,; F. Alhaidari, Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review. In Proceedings of the 2nd International Conference on Computer Applications and Information Security, ICCAIS 2019, Riyadh, SaudiArabia, 1–3 May 2019; pp. 1–7.

[18] A. Muraleedharan, N. Antony, R. Nandakumar, "Dynamic Time Slice Round Robin Scheduling Algorithm with Unknown Burst Time." Indian J. Sci. Technol. 2016, 9, 16.

[19] M. Hemamalini, , M.V. Srinath "Memory Constrained Load Shared Minimum Execution Time Grid Task Scheduling Algorithm in a Heterogeneous Environment." Indian J. Sci. Technol. 2015, 8.

[20] S. Negi, An Improved Round Robin Approach using Dynamic Time Quantum for Improving Average Waiting Time. Int. J. Comput. Appl. 2013, 69, 12–16.

[21] Feitelson, D.G.; Tsafrir, D.; Krakov, D. Experience with using the Parallel Workloads Archive. J. Parallel Distrib. Comput. 2014, 74, 2967–2982.

[22] Nasa-Workload. 2019. Available online: http://www.cs.huji.ac.il/labs/parallel/workload (accessed on 14 August 2019).

[23] Aida, K. Effect of job size characteristics on job scheduling performance. In Job Scheduling Strategies for Parallel Processing; Feitelson, D.G., Rudolph, L., Eds.; Lecture Notes in Computer Science; JSSPP 2000; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1911.

[24] Alboaneen, D.A.; Tianfield, H.; Zhang, Y. Glowworm swarm optimisation based task scheduling for cloud computing. In Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, Cambridge, UK, 22–23 March 2017; pp. 1–7.

[25] Chiang, S.H.; Vernon, M.K. Dynamic vs. Static quantum-based parallel processor allocation. In Workshop on Job Scheduling Strategies for Parallel Processing; JSSPP 1996; Lecture Notes in Computer Science; Feitelson, D.G., Rudolph, L., Eds.; Springer: Berlin/Heidelberg, Germany, 1996; Volume 1162.

[26] Yadav R., Mishra A., Prakash N., and Sharma H., "An Improved Round Robin Scheduling Algorithm for CPU Scheduling," International Journal on Computer Science and Engineering, vol. 2, no. 4, pp. 1064-1066, 2010.

[27] Chavan S., and P., and Tikekar., "An Improved Optimum Multilevel Dynamic Round Robin Scheduling Algorithm," International Journal of Scientific and Engineering Research, vol. 4, no. 12, pp. 298-301, 2013.

[28] Rajput I. and Gupta D., "A Priority Based Round Robin CPU Scheduling Algorithm for Real Time Systems," International Journal of Innovations in Engineering and Technology, vol. 1, no. 3, pp. 1- 11, 2012.