Received: July 16, 2022. Revised: August 8, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A New Round Robin Algorithm for Task Scheduling in Real-time System

Nermeen Ghazy[1]* **Afaf Abdelkader[1]** **Mervat S. Zaki[1]** **Kamal A. ElDahshan[2]**

[1]*Department of Mathematics, Faculty of Science (Girls), Al-Azhar University, Cairo, Egypt*
[2]*Department of Mathematics, Faculty of Science, Al-Azhar University, Cairo, Egypt*
* Corresponding author's Email: nermeena207@gmail.com

Abstract: The modern human life is strongly reliant on real-time systems. The operating system is one of the most significant real-time systems. To meet the demands of the future, operating systems must be upgraded. Nowadays, a lot of systems are used; like mobile applications and internet of things. Task scheduling is so important in the operating system to enhance the performance in these systems. Round Robin algorithm has been widely used in task scheduling. In this paper, a Median Mean Round Robin (MMRR) algorithm is proposed to significantly enhance the performance of the Round Robin algorithm. The proposed algorithm finds an optimal dynamic time quantum ⌊(median+ mean)/2⌋ and generated for each cluster depending on the remaining burst time of the processes. The performance has been enhanced in terms of waiting time, turnaround time and context switching. The experimental results show that the proposed algorithm outperforms ADRR, HYRR, EDRR and MARR algorithms.

Keywords: Task scheduling, CPU scheduling, Operating system, Round robin, Dynamic time quantum, MMRR.

1. Introduction

Real-Time Operating System (RTOS) is a reactive operating system that must respond to stimuli from a process it is attempting to regulate in real-time [1]. A reactive system that must meet time limitations is known as a real-time system [2]. A real-time system must be able to process information from the process in a timely manner without compromising process control[1]. The most important constraint to meet is time constraints. The validity of a real-time system is determined not only by the results of the treatment, but also by the temporal aspect. The real-time systems can be classified into three types [3]:

A. A real-time strict system: is one that is subject to strict time constraints, that is, one in which even the smallest time error can have fatal human and economic consequences. Most avionics, car, and other applications are strict real-time.

B. Real-time flexible system: a system that is subject to flexible time constraints and can accept a number of timing faults.

C. A real-time mixed system: is one in which time restrictions are both strict and flexible.

A real-time task is made up of a set of instructions that can be executed in order on one or more processors while keeping time limitations in mind. A real-time task can be:

A. Periodic: its instances (versions) are repeated indefinitely, and the duration between two consecutive activations of instances is constant (referred to as period).

B. Sporadic: its instances (versions) are repeated indefinitely, with a minimum time interval between them.

C. Aperiodic: no relationship exists between subsequent instances.

Scheduling is the most critical part of an operating system; it enables processes access to the system resources. Many requirements, such as fast computing, multitasking (running many processes at the same time), and multiplexing (transmitting numerous flows at the same time) [4]. It is too necessary to use of scheduling algorithms. When there are several runnable processes, scheduling is a fundamental function that chooses which one to run.

Some examples of scheduling algorithms like First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), and Priority Based Scheduling [5]. Except for Round Robin scheduling, the most of these algorithms are considered ineffective in real-time operating systems due to their poor performance. A number of assumptions are taken into account while scheduling CPUs, including the following:

A. A job pool is a collection of processes that are waiting for CPU time.

B. Each process is self-contained and competes for resources.

C. The scheduler's process is to distribute the CPU's limited resources across the many processes in a way that optimizes specific performance metrics.

The scheduler which is at the heart of the kernel, is responsible for determining which process should be run. An operating system can be classified into three sorts of schedulers in this context [6]: long-term which is load a process in the memory [7], mid-term or medium-term which is reduces memory consumption [7], and short-term which is select ready process to run on CPU [7]. Round Robin(RR) is a widely used scheduling algorithm that gives each process equal priority [8]. For the execution of the process, Round Robin uses a small unit of time called Time Quantum or Time Slice (TQ) [9]. If the CPU burst of a process exceeds 1-time quantum, the process is pre-empted and returned to the ready queue. A new process is added to the tail of the circular queue if it arrives. However, on a time-sharing operating system and real-time systems, RR performs better [10]. Round Robin algorithms have been tested using real-time operating systems, namely RTOS, and it has been discovered that these systems will work well since they are correctly configured to handle the scheduling process in real-time [11]. It uses a fixed time slice. All previous works based on Round Robin edit the time slice method. However, different approaches reveal distinct limitations [12]. When the time slice is too large, the processes in the ready queue become starved [13]. In the other side, the context switching time is high when TQ is small. RR improves response time and makes efficient use of shared resources. Due to the usage of static time quantum, processes with varying CPU bursts have longer waiting time, undesired overhead, and longer turnaround time. Using a dynamic time quantum with RR, it is automatically adapt to tasks in the queue. Although the current algorithms using a dynamic time quantum do not use several parameters in the selection of the quantum, which have an impact on the scheduling process and the system performances [14].

The main contribution of this paper is to improve the present Round Robin algorithm by enhancing the time quantum in real time for candidate processes without compromising its fairness. A new algorithm for changing time quantum in a progressive manner at various states of the ready queue is proposed. A mathematical model has been created to prove that the proposed algorithm outperforms the traditional Round Robin algorithm in terms of several performance metrics such as average waiting time, turnaround time and number of context switches. The proposed improved version of Round Robin algorithm outperforms the traditional Round Robin algorithm, according to the experimental results. This proposed algorithm solves the problem by using a progressive time quantum, which is modified repeatedly based on the remaining burst time of currently executing processes. Furthermore, the processes are sorted in ascending order of their burst time, and then the proposed algorithm is applied to each process to improve turnaround time, waiting time, and context switch. In comparison to the other RR techniques discussed in this work, the drawbacks of the discussed algorithms like ADRR [15], HYRR [16], EDRR [17] and MARR [18] are that they give a large average waiting time, average turnaround time and number of context switches. The contribution of this work is to: 1- Minimizes average waiting time and average turnaround time. 2- Reduce the number of context switches. The rest of the paper is organized as follows. In Section 2, scheduling criteria is reviewed. Section 3 focuses on some literature review. In Section 4, the proposed algorithm is presented in detail. Section 5 discusses the experimental results. In Section 6, result analysis is explained. The paper is concluded and described future work in Section 7.

2. Scheduling criteria

Each scheduling method has its own set of characteristics that aid in determining which scheduling algorithm will be more effective in the current problem [19].

2.1 CPU utilization

Best scheduling methods are those in which the CPU does not has a minimum idle time. Processing on the CPU must be occupied. Hard real-time CPU utilization is 89.9%, whereas soft real-time CPU utilization is 60% [20].CPU utilization time is the

percentage of time T spent by the CPU in the execution of processes.

2.2 Throughput

It is the number of processes that have been executed in a given amount of time. The CPU will provide good throughput if it is constantly busy.

2.3 Turnaround time

The total time it takes for the process to execute, from the time of arrival until the time of completion is called turnaround time [21] and calculated as in Eq. (1):

$$TAT_i = T_{cti} - T_{ati} \quad (1)$$

The average turnaround time (ATT) is calculated as in Eq.(2):

$$ATT = \frac{\sum_{i=0}^{n} TAT_i}{n} \quad (2)$$

Where TAT_i is the turnaround time of the processes, Tct_i is the completion time of the processes; $Tati$ is the arrival time of the processes, n is the number of the processes and ATT_i is the average turnaround time for the processes.

2.4 Response time

It is the period between the arrival of a process and the time at which it receives its first response (allocated to the CPU) [21].

2.5 Waiting time

It is the total time the process spent in ready queue [22]. It is calculated as in Eq. (3).

$$WT_i = T_{tati} - T_{bti} \quad (3)$$

The average waiting time (AWT) is calculated as in Eq. (4):

$$AWT = \frac{\sum_{i=0}^{n} WT_i}{n} \quad (4)$$

Where WTi is the waiting time of the processes, $Tbti$ is the burst time of the processes and AWT is the average waiting time for the processes.

2.6 Context switching

It is the process of shifting the CPU from one process to another. Switching time should be kept to a minimum because it wastes time during the process' execution [23].

Main objectives of a good scheduling algorithm are:

- Maximize CPU utilization.
- Maximize throughput.
- Minimize turnaround time.
- Minimize response time.
- Minimize waiting time.
- Minimize the number of context switching.

3. Literature review

For allocation processes, many CPU scheduling algorithms have been implemented. Taking the best features of each algorithm and combining them to create the ideal algorithm for a given situation.

In [24], the author proposed a modified genetic algorithm in cloud computing to identify the best servers to deploy these VMs and the best VMs to use for completing tasks that have been received. The chromosome of GAS is represented by this proposed algorithm using a matrix structure that integrates the ids of jobs, VMs, and servers. The algorithm achieved better performance in terms of makespan, scheduling length, throughput, resource utilization, energy consumption, and imbalance degree. In [25], an optimization model based on a Multi-Objective Improved Cuckoo Search Algorithm (MOICS) has been proposed in order to optimize task scheduling issues in a cloud environment as automatically allocating work to cloud nodes. This algorithm decreased both the processing time for the jobs and the overall cost0. Computational resources that can be used efficiently on cloud nodes are distributed using the proposed methodology. The proposed algorithm minimizes both makespan and cost. In [26], the author proposed a study that suggests task scheduling, resource mapping, data centre (DC) clustering, and virtual machine (VM) clustering to address the resource utilization and load imbalance that consumes larger waiting time of users. The region-based fuzzy probabilistic C-means clustering (R-FPCM) algorithm is initially used to cluster the DCs. Data dependencies, million instructions per second (MIPS), latency, storage, bandwidth, and VM counts are collected before performing a DC clustering operation. The VMs are clustered using multi-objective density-based spatial clustering based on estimated capacity and bandwidth depending on DC clustering. The Markov chain is used to predict future load and balance in accordance with it. To monitor the VM resources and map them according to the user's task requirements, an intermediate

broker is used. The broker satisfies the service level agreement (SLA) for each user before using a quick 1 to N resource mapping technique to map resources. The next step is an entropy-based monotonic scheduling algorithm that lists user tasks in the order specified by the task's size, type, deadline, and arrival time. Entropy computation that is performed dynamically makes it possible to optimize scheduling based on the state of the system. The system gave better results in terms of execution time, latency, resource utilization, and response time. In [27], In the cloud environment, a Hybrid Max-Min Genetic Algorithm (HMMGA) is offered for task scheduling and load balancing. Every Virtual Machine (VM) has its load initially measured; if the load is high, HMMGA is used to balance the load. The tasks are migrated from the over-loaded VMs to the under-loaded VMs by HMMGA, which chooses the optimal VMs to assign them to. In the cloud environment, HMMGA considerably reduces the performance imbalance caused by workload imbalance. In [28], the author proposed the Cuckoo Crow Search Algorithm (CCSA), an effective hybridized scheduling algorithm had been presented for improvising the task scheduling process. It mimics the parasitic behaviour of the cuckoo and the crow bird's habit of gathering food. The crow bird is always observing its neighbours in an effort to find a better food source than the one it has at the moment. At some situations the crow even goes a step further and snatches its neighbour’s food. The CCSA was created to be used in the cloud environment for finding an appropriate VM to execute the task scheduling process and was inspired by these traits of these birds. The proposed CCSA reduced makespan and cost.

In [29], the author proposed a modified Round Robin algorithm named average max Round Robin. In this algorithm, processes are scheduled for execution from the ready queue (RQ), implying that they have already been added to the ready queue. All processes in the ready queue have zero arrival time. The processes are sorted in an ascending order and the time quantum is calculated for the processes equal to (average + maximum burst time)/2. Processes are executed in iteration, and as the first iteration is completed, some processes are executed and then they deleted from the ready queue. The same process will be carried out until there are no more processes in the ready queue to run. Then the average waiting time and turnaround time are calculated. In [30], a new median Round Robin algorithm has been proposed named modified median Round Robin algorithm (MMRRA). The authors use a dynamic TQ calculated by taking the square root of median and highest burst time of the process. This algorithm includes an essential condition. If any process completes its first cycle of time quantum and the remaining burst time of this process is greater than 20% of its total burst time, then those processes will go for the second cycle of the processing; otherwise, the CPU will complete the processes. In [16], an improvement is applied to the Round Robin algorithm named an efficient customized Round Robin algorithm (EDRR) by choosing a dynamic time quantum that would let a process to complete if the remaining execution time was less than or equal to 0.2th of the total execution time. The maximum burst time is founded from the available processes in the ready queue. The time quantum is then calculated as a percentage of this time which is a 0.8th fraction of the maximal burst time. The scheduler now allocates the CPU to all processes in the ready queue that have a burst time smaller than the time quantum, while the bigger ones are held in reserve. The time quantum is set to maximal burst time once all of the smaller processes have completed their execution. This algorithm improves the system's performance by reducing on average turnaround and waiting times.

In [1], an improved Round Robin scheduler is developed named priority based Round Robin (PBRR) CPU scheduling algorithm. With a small enhancement, it is close to RR. It takes priority into account based on task management. Each process is assigned a priority index, and then the processes are sorted according to priority index in the ready queue. The first process in the ready queue is selected by this algorithm and the CPU is allocated for a time interval of time quantum. The allocated processes for a time interval that they have been executed are placed at the end of the ready queue. When processes have finished execution, they removed from the ready queue and the average of waiting time, turnaround time and response time are calculated. In [15], a modified Round Robin scheduler is developed named a novel amended dynamic Round Robin Scheduling algorithm (ADRR). The authors used dynamic time quantum to modify the traditional RR algorithm. The TQ, which is a critical element of RR's performance, is set to the value of the lowest CPU burst time. The authors set a TQ threshold of 20 and then check a condition: if TQ is less than the threshold (20), the condition is true, and TQ is set to 20. This condition is checked to ensure that the Value of TQ does not become too little small, resulting in an increased number of context switches. TQ is re-adjusted after each cycle. All processes are sorted in the ready queue in an increasing order based on the CPU burst

time. They are assigned to the CPU for a time interval. If a process's remaining CPU burst time exceeds half of the TQ, it will be pre-empted. Pre-empted processes are reinserted in an ascending order into the ready queue. The same principle applies until all of the processes are finished. In [17], the authors performed an innovative scheduling algorithm to reduce the average of waiting time, turnaround time, response time and number of context switches. It is called a hybrid Round Robin scheduling mechanism (HYRR) for process management. The mean of burst time and the minimum of burst time are used to calculate time quantum dynamically. Enhanced time quantum (ETQ) is calculated in phase 1 using the mean and lowest burst time. Following the calculation, the process with the shortest burst time and which is not currently being run in the CPU is assigned high priority and allocated for 1 Enhanced quantum time in the CPU. Phase-1 is performed until each process gets a single CPU allocation. Phase 2 places ready queue processes in ascending order based on their remaining burst time. Following the arrangement, the first process in the ready queue is given 1 quantum time in the CPU. If the current executing process's burst time in the CPU is less than or equal to 1ETQ, then the current process is reallocated in the CPU. The second phase is performed until the ready queue is empty. In [22], the authors proposed a multi-programmed operating system's Round Robin algorithm. The authors enhanced the value of time quantum by partition the ready queue into three sub queues: highest, medium, and lowest priority. To assign the value of time quantum to one of these sub-queues, it depends on a threshold value. This algorithm uses a separated time quantum for every sub-queue. All processes in each sub-queue should be finished execution respectively. This algorithm has been reduced drastically the results in metrics of average waiting time and turnaround time. In [18], the authors developed a new Round Robin algorithm by using a dynamic time quantum. They use the average and median of the burst time for each process (MARR). This algorithm enhances the average waiting time and turnaround time.

All of the improvements to Round Robin CPU scheduling described above are have some issues. The processes that enter the system may have a varying burst time, which means that their CPU execution time can vary. If all of the processes are sent to the CPU for execution in an increasing order, it will help to improve the turnaround time and waiting time. The RR algorithm operates on a fixed time quantum (TQ). The Round Robin algorithm has two possible outcomes: quantum time is either high or low. If the time quantum is large, the round-robin algorithm will run on a first-come-first-serve (FCFS) basis, and if the time quantum is extremely low, the algorithm will fail and produce a large number of context switches. So this paper proposes an optimal time quantum which is dynamic that solves this problem which enhances the performance of the system by: maximizing CPU utilization, minimizing waiting time and turnaround time and reducing number of context switches [31].

4. Proposed algorithm (median mean round robin algorithm)

This approach of RR scheduling algorithm solves the drawbacks of the Round Robin algorithm. In processes with small execution time, the Round Robin algorithm is not efficient because it provides a large number of context switches. Therefore, the waiting time and response time of process increase and hence the throughput of the system is decrease. In this proposed algorithm, we have implemented RR algorithm while taking the mean and the median of burst time of the processes into account. If all of the processes arrive in the ready queue at the same time, they are arranged in an ascending order based on their burst time. Then the time quantum is calculated dynamically by using the median and the mean as in Eq. (5).

$$TQ = \lfloor \left(\frac{(median+mean)}{2}\right) \rfloor \quad (5)$$

Where TQ is the time quantum of processes, median is median of burst time of all processes as in Eqs. (6) and (7) and mean is the summation of all processes divided by the number of all processes as in Eq. (8).

$$Med(BTi) = BTi\left[\frac{n+1}{2}\right] \quad \text{if n is odd} \quad (6)$$

$$Med(BTi) = [BTi(\frac{n}{2})] + [BTi\left(\frac{n}{2}\right) + 1]/2 \quad \text{if n is even} \quad (7)$$

$$mean = \frac{\sum_{i=1}^{n} BTi}{n} \quad (8)$$

Where BTi is the burst time of the process, n is the number of all processes. After calculation, the first process in the ready queue is allocated in the CPU. If the remaining CPU's burst time of the currently running process is less than half of the time quantum, the CPU is reallocated again to the currently running process for the remaining CPU burst time. Otherwise, the process will be terminated

to the tail of the ready queue. After allocation, if all processes are completed its execution and the ready queue is empty, then the average of waiting time and turnaround time is calculated. The problem of higher average waiting time, turnaround time and the large number of context switches are solved. Hence, the proposed algorithm enhances the performance of the system. The flowchart of the proposed algorithm is described as follow in Fig. 1.

Median Mean Round Robin algorithm:

1. Assign processes to the ready queue.
2. Arrange all the processes in an ascending order according to their burst time
3. TQ ← ⌊(median + mean) / 2⌋
4. While (ready queue! =NULL)
5. If (remaining burst time < 0.5 * TQ)
6. Allocate CPU again to the current running process for the remaining burst time.

   Else

7. Put the remaining of the current process at the end of the ready queue.
8. Go to step 4
9. End while
10. Calculate average waiting time, average turnaround time and the number of context switches (NOS).

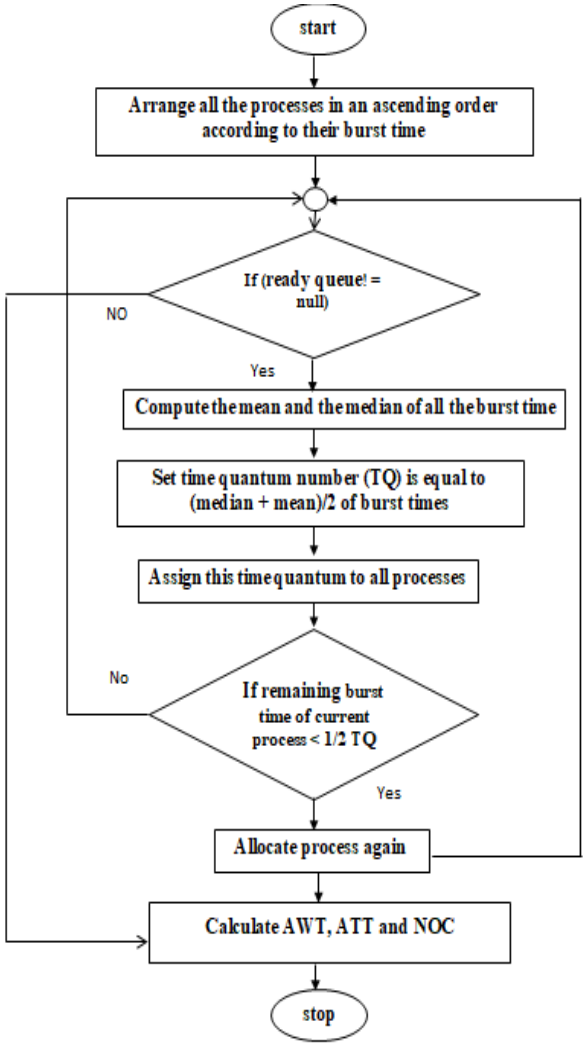


Figure. 1 Flowchart of the MMRR algorithm

5. Experimental results

In this section, we described the results of the proposed algorithm to prove the effectiveness of it. We choose cases utilised in the majority of these algorithms, with the same number of processes, burst time, and arrival time, to demonstrate the effectiveness of our technique and to ensure a fair comparison between the proposed algorithm and the current algorithms. P1, P2, P3, P4, and P5 are the five processes that have been considered.

5.1 Case 1: process are in random

The processes are arriving at zero time with random burst time 42, 101, 135, 68 and 170 respectively. Table 1 shows the burst time of processes with zero arrival time. Fig. 2 to 7 show the Gantt charts for the algorithms; RR [8], ADRR [15], HYRR [17], EDRR [16], MARR [18] and the proposed algorithm (MMRR).

5.1.1. RR

The processes are entered in the ready queue in their order; P1=42, P2=101, P3=135, P4=68 and P5=170. Let TQ= 40.

Table 1. Processes are coming in random order

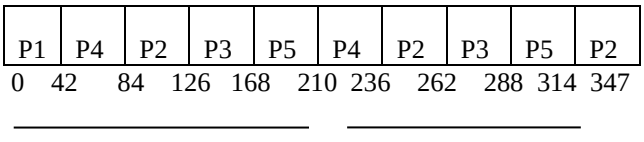
Process	Burst time
P1	42
P2	101
P3	135
P4	68
P5	170

P1	P2	P3	P4	P5	P1	P2	P3	P4
0	40	80	120	160	200	202	242	282	310

P5	P2	P3	P5	P3	P5	P5
310	350	371	411	451	466	506	516

Figure. 2 Gantt chart for RR algorithm (case 1)

AWT= 269.8ms
ATT = 373ms



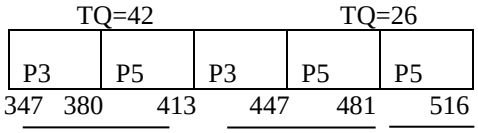


Figure. 3 Gantt chart for ADRR algorithm (case 1)

AWT= 214.4ms
ATT = 317.6ms

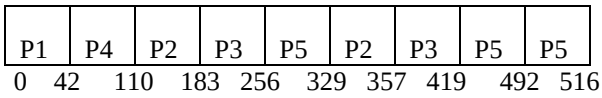


Figure. 4 Gantt chart for HYRR algorithm (case 1)

AWT= 185.6ms
ATT = 288.8ms

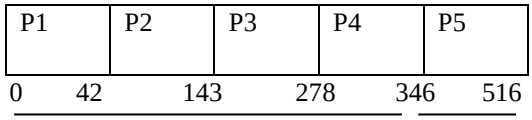


Figure. 5 Gantt chart for EDRR algorithm (case 1)

AWT= 161.8ms
ATT = 265ms

5.1.2. ADRR

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=42, P4=68, P2=101, P3=135 and P5=170. TQ is calculated according to ADRR algorithm. The calculated TQ is 42, 26, 33, 34 and 35.

5.1.3. Hybrid round robin (HYRR)

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=42, P4=68, P2=101, P3=135 and P5=170. TQ is calculated according to HYRR algorithm. The calculated TQ is 73.

5.1.4. EDRR

Enter the processes in the ready queue in their order; P1=42, P2=101, P3=135, P4=68 and P5=170. TQ is calculated according to EDRR algorithm. The calculated TQ is 136 and 170.

5.1.5. MARR

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=42, P4=68, P2=101, P3=135 and P5=170. TQ is

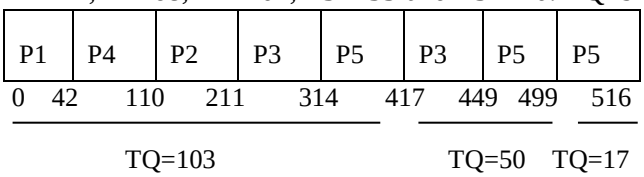


Figure. 6 Gantt chart for MARR algorithm (case 1)

AWT= 162.2ms
ATT = 265.4ms

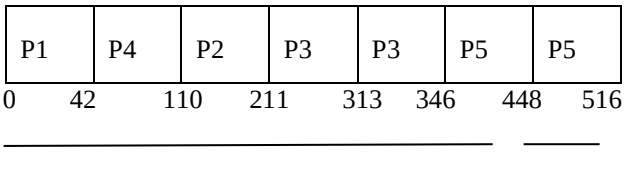


Figure. 7 Gantt chart for median mean round robin algorithm (case 1)

AWT= 141.8ms
ATT = 245ms

calculated according to MARR algorithm. The calculated TQ is 102, 50 and 18.

5.1.6. Proposed algorithm (median mean round robin)

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=42, P4=68, P2=101, P3=135 and P5=170. TQ is calculated according to proposed algorithm. The calculated TQ is 102 and 68.

The following Table 2 presents a comparative study among the existing algorithms with respect to TQ, AWT, ATT and number of context switches for case 1.

Fig. 8 below shows the comparison of average waiting time, average turnaround time and NOS for the existing algorithms.

5.2 Case 2: The incoming burst time in an increasing order

The processes are arriving at zero time with increasing burst time 5, 45, 78, 90 and 120

Table 2. Comparative study of RR, ADRR, HYRR, EDRR, MARR and MMRR algorithms (case 1)

Algorithm	TQ	AWT (ms)	ATT (ms)	NOS
RR	40	269.8	373	14
ADRR	42,26,33, 34,35	214.4	317.6	13
HYRR	73	185.6	288.8	7

EDRR	136,170	161.8	265	4
MARR	103,50,17	162.4	265.6	6
proposed	49,50	141.8	245	4

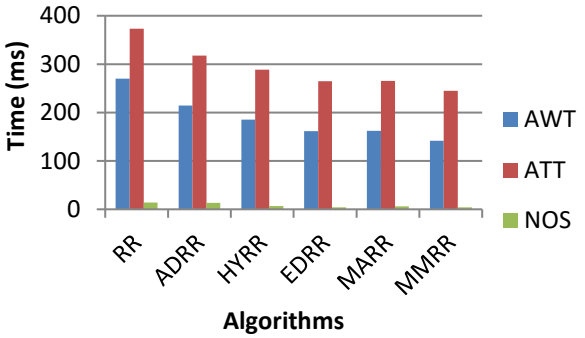


Figure. 8 Comparative graph for the average waiting time, turnaround time and the NOS (case 1)

Table 3. Processes are in increasing order

Process	Burst time
P1	5
P2	45
P3	78
P4	90
P5	120

respectively are shown in Table 3. Fig. 9 to 14 show the Gantt charts for the algorithms; RR[8], ADRR[15], HYRR[17], EDRR[16], MARR[18] and the proposed algorithm (MMRR).

5.2.1. RR

The processes are entered in the ready queue in their order; P1=5, P2=45, P3=78, P4=90 and P5=120. Let TQ=25.

5.2.2. ADRR

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=5, P2=45, P3=78, P4=90 and P5=120. TQ is calculated according to ADRR algorithm. The calculated TQ is 20, 25, 33 and 42.

5.2.3. HYRR

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=5, P2=45, P3=78, P4=90 and P5=120. TQ is calculated according to HYRR algorithm. The calculated TQ is 37.

5.2.4. EDRR

The processes are entered in the ready queue in their order; P1=5, P2=45, P3=78, P4=90 and P5=120 according to TQ. TQ is calculated according to EDRR algorithm. The calculated TQ is 96 and 120.

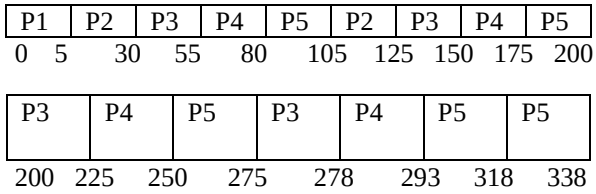


Figure. 9 Gantt chart for RR (case 2)

AWT= 140.2ms
ATT = 207.8ms

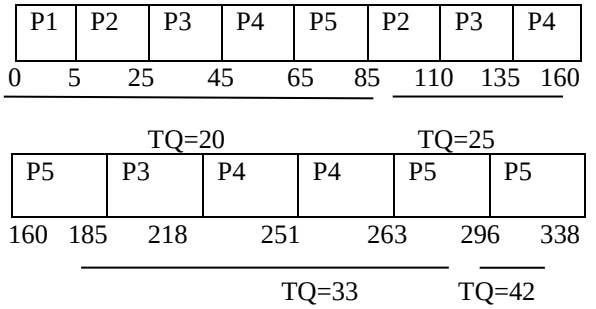


Figure. 10 Gantt chart for ADRR (case 2)

AWT= 119.2ms
ATT = 186.8ms

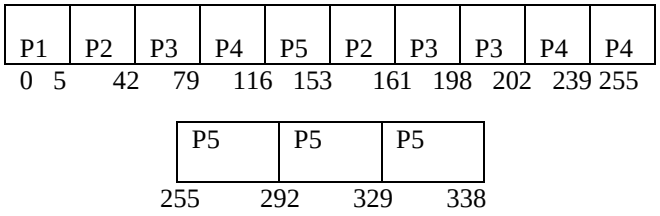


Figure. 11 Gantt chart for HYRR algorithm (case 2)

AWT= 124.6ms
ATT = 192.2ms

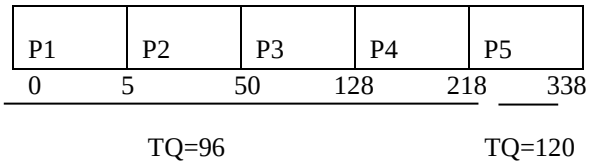


Figure. 12 Gantt chart for EDRR algorithm (case 2)

AWT= 80.2ms
ATT = 147.8ms

5.2.5. MARR

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=5, P2=45, P3=78, P4=90 and P5=120. TQ is calculated according to MARR algorithm. The calculated TQ is 72, 21 and 27.

5.2.6. Proposed algorithm (median mean round robin)

All the processes are sorted in the ready queue in increasing order according to their burst time; P1=5, P2=45, P3=78, P4=90 and P5=120. Calculate the time quantum according to MMRR algorithm. The calculated time quantum is 72 and 48.

The following Table 4 presents a comparative study among the existing algorithms with respect to TQ, AWT, ATT and number of context switches for case 2.

Fig.15 below shows the comparison of average waiting time, average turnaround time and number of context switches for the existing algorithms.

It is noticed that from the results, EDRR algorithm gave the same results of the average waiting time and turnaround time as the proposed algorithm in the case of the processes coming in increasing order because they are already arranged.

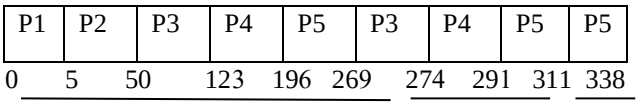


Figure. 13 Gantt chart for MARR algorithm (case 2)

AWT= 124ms
ATT = 191.6ms

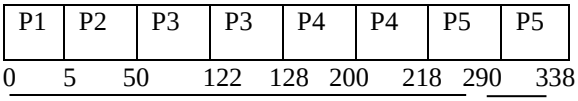


Figure. 14 Gantt chart for median mean round robin algorithm (case 2)

AWT= 80.2ms
ATT = 147.8ms

Table 4. Comparative study of RR, ADRR, HYRR, EDRR, MARR and proposed algorithm (case 2)

Algorithm	TQ	AWT (ms)	ATT (ms)	NOS
RR	25	140.2	207.8	14
ADRR	20,25, 33,42	119.2	186.8	11
HYRR	37	102.4	192.2	8
EDRR	96,120	80.2	147.8	4
MARR	73,20, 27	124	191.6	7
proposed	72,48	80.2	147.8	4

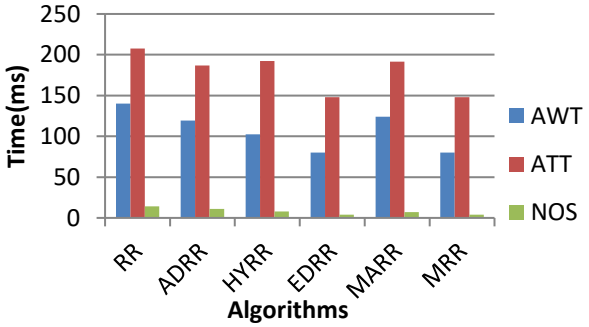


Figure. 15 Comparative graph for the average waiting time, turnaround time and the NOS (case 2)

5.3 Case 3: Process are coming in decreasing order

The processes are arriving at zero time with decreasing burst time 140, 78, 62, 45 and 34 respectively are shown in Table 5.

Fig.16 to 21 show the Gantt charts for the algorithms; RR [8], ADRR [15], HYRR [17], EDRR [16], MARR [18] and the proposed algorithm(MMRR).

5.3.1. RR

The processes are entered in the ready queue in their order; P1=140, P2=78, P3=62, P4=45 and P5=34. Let TQ= 30.

5.3.2. ADRR

All the processes are sorted in the ready queue in increasing order according to their burst time; P5=34, P4=45, P3=62, P2=78 and P1=140. TQ is calculated according to ADRR algorithm. The calculated TQ is 34, 28, 20 and 58.

Table 5. Processes are in decreasing order

Process	Burst time
P1	140
P2	78
P3	62
P4	45
P5	34

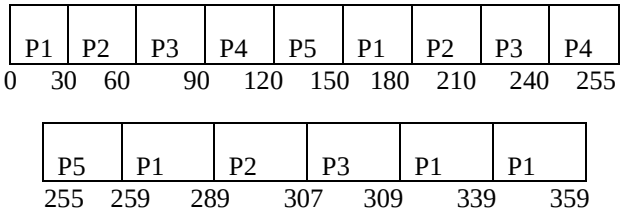


Figure. 16 Gantt chart for RR algorithm (case 3)

AWT= 226ms
ATT = 297.8ms

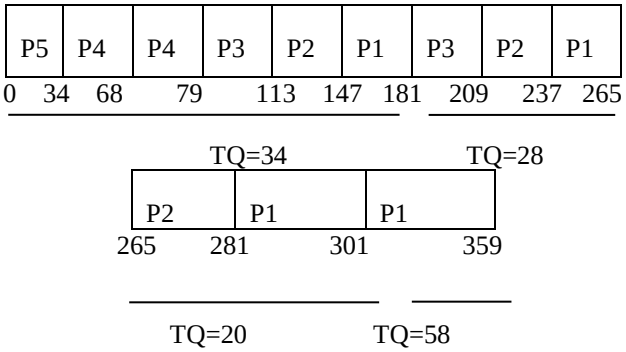


Figure. 17 Gantt chart for ADRR algorithm (case 3)

AWT= 120.6ms
ATT = 192.4ms

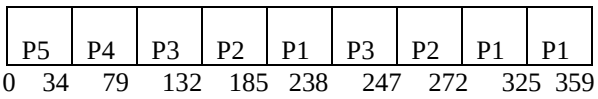


Figure. 18 Gantt chart for HYRR algorithm (case 3)

AWT= 126.4ms
ATT = 198.2ms

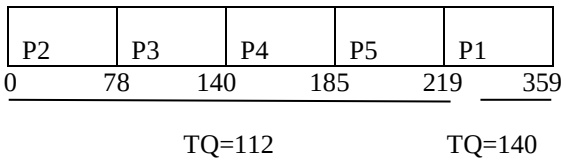


Figure. 19 Gantt chart for EDRR algorithm (case 3)

AWT= 124.4ms
ATT = 196.2ms

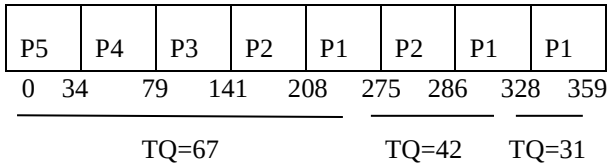


Figure. 20 Gantt chart for MARR algorithm (case 3)

AWT= 107.8ms
ATT = 179.6ms

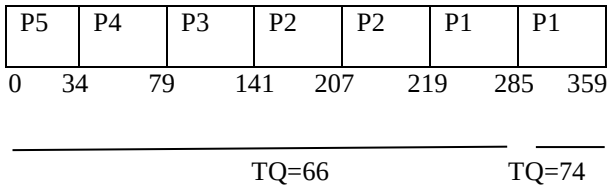


Figure. 21 Gantt chart for median mean round robin algorithm (case 3)

AWT= 65.2ms
ATT = 121.2ms

5.3.3. HYRR

All the processes are sorted in the ready queue in increasing order according to their burst time; P5=34, P4=45, P3=62, P2=78 and P1=140. TQ is calculated according to HYRR algorithm. The calculated TQ is 53.

5.3.4. EDRR

The processes are entered in the ready queue in their order according to time quantum; P1=140, P2=78, P3=62, P4=45 and P5=34. TQ is calculated according to EDRR algorithm. The calculated TQ is 112, 140.

5.3.5. MARR

All the processes are sorted in the ready queue in increasing order according to their burst time; P5=34, P4=45, P3=62, P2=78 and P1=140. TQ is calculated according to MARR algorithm. The calculated TQ is 66, 43 and 31.

5.3.6. Proposed algorithm (median mean round robin)

All the processes are sorted in the ready queue in increasing order according to their burst time; P5=34, P4=45, P3=62, P2=78 and P1=140. TQ is calculated according to proposed algorithm. The calculated TQ is 66, 74.

The following Table 6 presents a comparative study among the existing algorithms with respect to TQ, AWT, ATT and number of context switches for case 3.

Fig. 22 below shows the comparison of average waiting time, average turnaround time and number of context switches for the existing algorithms.

6. Result analysis

Proposed algorithm (MMRR) is compared with Amended Dynamic Round Robin (ADRR) [15], Hybrid Round Robin (HYRR) [17], An efficient Customized Round Robin (EDRR) [16] and Median average Round Robin (MARR) [18]. All of these algorithms are compared with Round Robin algorithm [8] in order to evaluate their performance. Proposed algorithm (MMRR), Round Robin, and

Table 6. Comparative study of RR, ADRR, HYRR, EDRR, MARR and proposed algorithm (case 3)

Algorithm	TQ	AWT (ms)	ATT (ms)	NOS
RR	30	226	297.8	13
ADRR	34, 28, 20, 58	120.6	192.4	9
HYRR	53	126.4	198.2	7
EDRR	112, 140	124.4	196.2	4
MARR	67, 42, 31	108	179.8	6
proposed	66,74	94.6	166.4	4

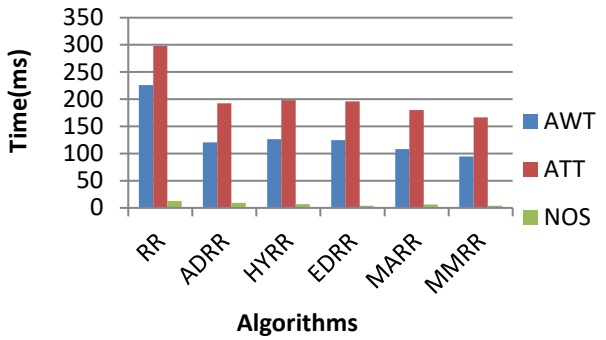


Figure. 22 Comparative graph for the average waiting time, turnaround time and the NOS (case 3)

other algorithms are implemented in C ++ and compared using the same random data set. The comparison of these algorithms is based on average waiting time and average turnaround time. Because of the number of processes in the ready queue determines average waiting and turnaround time, an increase in time results to a rise in cost. The experimental data used two different data sets from (10-100) and from (500-5000) processes. The comparison of algorithms in terms of average waiting time is shown in Fig.23 and Fig 25. For (10 to 100) and (500 to 5000) processes, the stacked line chart is plotted lying in the ready queue. The average waiting time of the processes is provided in milliseconds and plotted against the y-axis, while the number of processes in the ready queue is plotted against the x-axis. The proposed algorithm (MMRR) gives better results, followed by EDRR [16], MARR [18], HYRR [17] and ADRR [15]. A substantial improvement is given by these algorithms when compared to RR algorithm. As the number of processes is increasing in the ready queue the performance of the algorithms is enhanced. The EDRR, MARR, HYRR gives significant results compared to RR while ADRR gives reasonable improvement results compared to RR. Whereas the proposed algorithm shows more significant improvement results than other algorithms. In terms of a lower number of processes, the proposed algorithm act similarly, however as the number of processes increases, the performance of MMRR showed an upward trend in average waiting time compared to other algorithms. In comparison to suggested algorithms, the average waiting time for RR is consistently increasing, as seen in the line chart.

The behaviour of algorithms in terms of average turnaround time exhibits a similar pattern as shown in Fig 24 and Fig 26. For (10 to 100) and (500 to 5000) processes, the stacked line chart is plotted lying in the ready queue. The average turnaround time of the processes is provided in milliseconds and plotted against the y-axis, while the number of processes in the ready queue is plotted against the x-axis. The proposed algorithm (MMRR) gives better results, followed by EDRR [16], MARR [18], HYRR [17] and ADRR [15]. A substantial improvement is given by these algorithms when compared to RR algorithm. As the number of processes is increasing in the ready queue the performance of the algorithms is enhanced. The EDRR, MARR, HYRR gives significant results compared to RR while ADRR gives reasonable improvement results compared to RR. In terms of a lower number of processes, the proposed algorithms act similarly, however as the number of processes increases, the performance of MMRR showed an upward trend in average turnaround time compared to other algorithms. In comparison to suggested algorithms, the average turnaround time for RR is consistently increasing, as seen in the line chart.

It is obvious that the proposed algorithm is efficient and effective for CPU process scheduling.

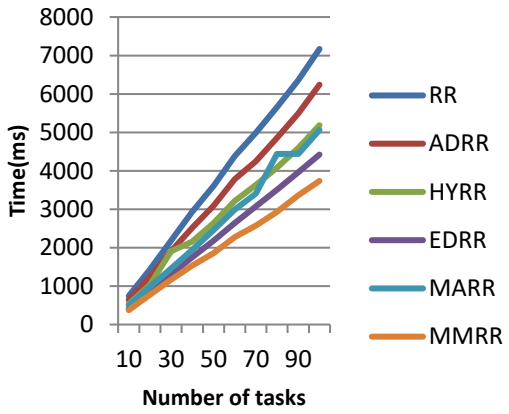


Figure. 23 Comparative graph for the average waiting time

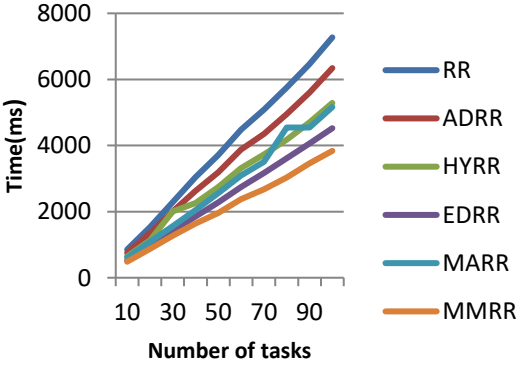


Figure. 24 Comparative graph for the average turnaround time

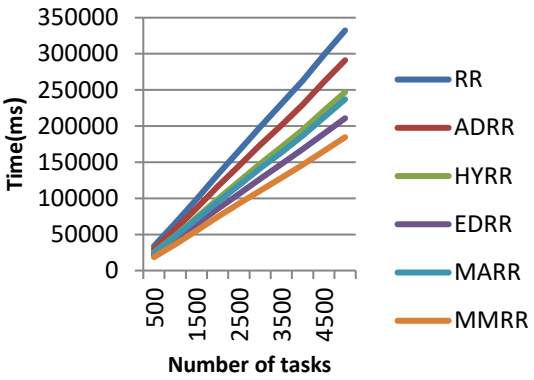


Figure. 25 Comparative graph for the average waiting time

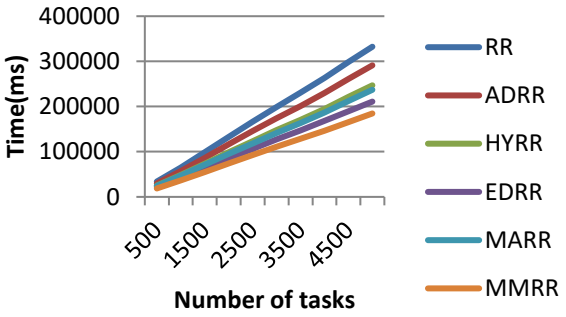


Figure. 26 Comparative graph for the average turnaround time

7. Conclusion and future work

Many improvements have been developed the Round Robin algorithm for task scheduling to be compatible with time sharing systems and real-time systems. The most important issue in the RR algorithm is the time quantum. The main contribution of this paper is enhancing the efficiency of the RR algorithm by proposing a Median Mean Round Robin algorithm (MMRR). It found an intelligent dynamic time quantum which is calculated as ⌊(median +mean)/2⌋ and taking the remaining burst time of the current process into account. If the remaining burst time of the current process is less than half of the time quantum, it allocated again. Otherwise, it is placed at the end of the ready queue. Each cycle will have its TQ based on their burst time for these processes. By using a variable TQ depends on the burst time, it lead to minimize the average waiting time, turnaround time and number of context switches. The experimental results showed that the proposed algorithm enhances the performance of the system by reducing the average waiting time, turnaround time and number of context switches. The proposed MMRR algorithm when compared to the RR, ADRR, HYRR, EDRR and MARR algorithms, successfully optimised the average waiting time, turnaround time and number of context switches. So this algorithm meets the demands in real-time systems. In the future work, we will improve the RR algorithm by developing an algorithm of time quantum calculation combining the dynamic and fixed quantum values.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization of this paper, Kamal, Mervat, Afaf and Nermeen; methodology, Kamal, Afaf, and Nermeen; the software, Afaf and Nermeen; writing (original draft), Nermeen; review and editing, Kamal, Afaf and Nermeen.

Acknowledgments

The authors thank the editors and the anonymous reviewers for their valuable suggestions.

References

[1] S. Zouaoui, L. Boussaid, and A. Mtibaa, “Priority based round robin (PBRR) CPU scheduling algorithm”, *International. Journal of Electrical and Computer Engineering*, Vol. 9, No. 1, pp. 190-202, 2019.

[2] P. Patra and P. Pradhan, “An integrated dynamic model optimizing the risk on real time operating system”, *International Journal of Information Security and Privacy*, Vol. 8, No. 1, pp. 38-61, 2014.

[3] M. Awadalla, “Heuristic Approach for Scheduling Dependent Real-Time Tasks”, *Bulletin of Electrical Engineering and Informatics*, Vol. 4, No. 3, pp. 217-230, 2015.

[4] S. Iqbal, H. Gull, S. Saeed, M. Saqib, M. Alqahtani, Y. Bamarouf, G. Krishna, and M. Aldossary, “Relative time quantum-based enhancements in Round robin scheduling”, *Computer Systems Science and Engineering*, Vol. 41, No. 2, pp. 461-477, 2022.

[5] K. Eldahshan, A. Abdelkader, and N. Ghazy, “Round Robin based Scheduling Algorithms, A Comparative Study”, *Automatic Control and System Engineering Journal*, Vol. 17, No. 2, pp. 29-42, 2017.

[6] N. Harki, A. Ahmed, and L. Haji, “CPU Scheduling Techniques: A Review on Novel Approaches Strategy and Performance Assessment”, *Journal of Applied Science and Technology Trends*, Vol. 1, No. 2, pp. 48-55,

2020.

[7] R. Mishra and G. Mitawa, "Scheduling Process for CPU", In: *Proc. of Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks*, pp. 590-593, 2021.

[8] S. Mostafa and H. Amano, "Dynamic round robin CPU scheduling algorithm based on K-means clustering technique", *Applied Sciences (Switzerland)*, Vol. 10, No. 15, 5134, 2020.

[9] B. Richardson and W. Istiono, "Comparison Analysis of Round Robin Algorithm with Highest Response Ratio Next Algorithm for Job Scheduling Problems", *International Journal of Open Information Technologies*, Vol. 10, No. 2, pp. 21-26, 2022.

[10] P. Banerjee, B. Kumar, and P. Banerjee, "Mixed Round Robin Scheduling for Real Time Systems", *International Journal of Computer Trends and Technology*, Vol. 49, No. 3, pp. 189-195, 2017.

[11] Hayatunnufus, M. Riasetiawan, and A. Ashari, "Performance Analysis of FIFO and Round Robin Scheduling Process Algorithm in IoT Operating System for Collecting Landslide Data", In: *Proc. of International Conference on Data Science, Artificial Intelligence, and Business Analytics*, pp. 63-68, 2020.

[12] T. Balharith and F. Alhaidari, "Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review", In: *Proc. of 2nd International Conference on Computer Applications & Information Security*, pp. 1-7, 2019.

[13] S. Mostafa and H. Amano, "An adjustable variant of round robin algorithm based on clustering technique", *Computers, Materials and Continua*, Vol. 66, No. 3, pp. 3253-3270, 2020.

[14] A. Fiad, Z. Maaza, and H. Bendoukha, "Improved version of round robin scheduling algorithm based on analytic model", *International Journal of Networked and Distributed Computing*, Vol. 8, No. 4, pp. 195-202, 2020.

[15] U. Shafi, M. Shah, A. Wahid, K. Abbasi, Q. Javaid, M. Asghar, and M. Haider, "A novel amended dynamic round robin scheduling algorithm for timeshared systems", *The International Arab Journal of Information Technology*, Vol. 17, No. 1, pp. 90-98, 2020.

[16] P. Sharma and Y. Sharma, "An Efficient Customized Round Robin Algorithm for CPU Scheduling", In: *Proc. of the Second International Conference on Information Management and Machine Intelligence*. *Springer*, pp. 623-629, 2021.

[17] K. Faizan, A. Marikal, and K. Anil, "A Hybrid Round Robin Scheduling Mechanism for Process Management", *International Journal of Computer Applications*, Vol. 177, No. 36, pp. 14-19, 2020.

[18] Sakshi, C. Sharma, S. Sharma, S. Kautish, S. Alsallami, E. Khalil, and A. Mohamed, "A new median-average round Robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time", *Alexandria Engineering Journal*, Vol. 61, No. 12, pp. 10527-10538, 2022.

[19] T. Paul, R. Hossain, and M. Samsuddoha, "Improved Round Robin Scheduling Algorithm with Progressive Time Quantum", *International Journal of Computer Applications*, Vol. 178, No. 49, pp. 30-36, 2019.

[20] P. Sharma, S. Kumar, M. Gaur, and V. Jain, "A novel intelligent round robin CPU scheduling algorithm", *International Journal of Information Technology*, Vol. 14, No. 3, pp. 1475-1482, 2021.

[21] K. E. Dahshan, A. Abdelkader, and N. Ghazy, "Achieving Stability in the Round Robin Algorithm", *International Journal of Computer Applications*, Vol. 172, No. 6, pp. 15-20, 2017.

[22] K. Arif, M. Morad, M. Mohammed, and M. Subhi, "An Efficient Threshold Round-Robin Scheme for CPU Scheduling (ETRR)", *Journal of Engineering Science and Technology*, Vol. 15, No. 6, pp. 4048-4060, 2020.

[23] A. Gupta, P. Mathur, C. T. Gonzalez, M. Garg, and D. Goyal, "ORR: Optimized Round Robin CPU Scheduling Algorithm", In: *Proc. of the International Conference on Data Science, Machine Learning and Artificial Intelligence*, pp. 296-304, 2021.

[24] A. Emara, A. G. Elrab, A. Sobhi, and K. Raslan, "Genetic-Based Multi-objective Task Scheduling Algorithm in Cloud Computing Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 5, pp. 571-582, 2021, doi: 10.22266/ijies2021.1031.50.

[25] S. Jaber, Y. Ali, and N. Ibrahim, "An Automated Task Scheduling Model Using a Multi-objective Improved Cuckoo Optimization Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 1, pp. 295-304, 2022, doi: 10.22266/ijies2022.0228.27.

[26] J. Varghese and J. Sreenivasaiah, "Entropy Based Monotonic Task Scheduling and

Dynamic Resource Mapping in Federated Cloud Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 1, pp. 235-250, 2022, doi: 10.22266/ijies2022.0228.22.

[27] S. Kodli and S. Terdal, "Hybrid Max-Min Genetic Algorithm for Load Balancing and Task Scheduling in Cloud Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 1, pp. 63-71, 2020, doi: 10.22266/ijies2021.0228.07.

[28] P. Krishnadoss, G. Natesan, J. Ali, M. Nanjappan, P. Krishnamoorthy, and V. K. Poornachary, "CCSA: Hybrid Cuckoo Crow Search Algorithm for Task Scheduling in Cloud Computing", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 4, pp. 241-250, 2021, doi: 10.22266/ijies2021.0831.22.

[29] P. Banerjee, P. Banerjee, and S. 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*, No. 1, pp. 2278-3075, 2012.

[30] H. Mora, S. Abdullahi, and S. Junaidu, "Modified Median Round Robin Algorithm (MMRRA)", In: *Proc. of 13th International Conference on Electronics, Computer and Computation*, pp. 1-7, 2017.

[31] S. Ali, R. Alshahrani, A. Hadadi, T. Alghamdi, F. Almuhsin, and E. E. Sharawy, "A Review on the CPU Scheduling Algorithms : Comparative Study", *International Journal of Computer Science & Network Security*, Vol. 21, No. 1, pp. 19-26, 2021.