Received: September 12, 2022. Revised: September 28, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

MMMRR: a Modified Median Mean Round Robin Algorithm for Task Scheduling

Afaf Abdelkader[1] Nermeen Ghazy[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: Real-time task scheduling is a critical part of systems since processes must finish their work at a certain time. Large data streams require high energy efficiency and rapid response times. Real-time system performance must be improved through task scheduling on resources. Numerous researchers have recently and rapidly proposed enormous efforts with enhancements in numerous task scheduling methods. Round Robin algorithm has been commonly utilized in task scheduling. In this paper, a Modified Median Mean Round Robin (MMMRR) algorithm is proposed to significantly enhance the performance of the Round Robin algorithm. The proposed algorithm finds an ideal dynamic time quantum $\lfloor(\text{median}+\text{mean})/2\rfloor$ and generated for each round depending on the remaining burst time of the processes. The system 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, MARR and MMRR algorithms.

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

1. Introduction

Scheduling is a mechanism for allocating resources to particular tasks in order to carry them out successfully. The main goals of scheduling are to:

1) maximize resource utilization while minimizing makespan, 2) maximize resource utilization, 3) maximize server utilization for tasks and 4) execute activities with higher priority while reducing overall average waiting time and completion time [1]. Moreover, improving performance and increasing system throughput are the primary benefits of effective scheduling. Task scheduling is the process of allocating incoming tasks in a specific way to make the most use of the available resources. In systems without scheduling, waiting times for jobs may be higher, and some short-term tasks may fail as a result of the delay.

At the time of scheduling, the scheduler must take into account a number of restrictions, including the task's nature, size, execution time, resource availability, task queue, and resource load. Task scheduling is one of the major problems in real-time systems. The effective use of resources may result from proper task scheduling [2].

The usage of scheduling algorithms is extremely important. When there are multiple runnable processes, scheduling is an important function that determines which process to start. First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), and Priority Based Scheduling are a few examples of scheduling algorithms[1]. With the exception of Round Robin scheduling, the most of these algorithms are thought to be ineffective in real-time systems due to their poor performance[1].

Round Robin (RR) is a commonly used scheduling algorithm. It gives every process the same priority [2]. It uses a small period of time known as a time quantum (TQ) to execute the process[3]. A process is pre-empted and put back on the ready queue if its CPU burst exceeds one time quantum. If a new process enters, it is added to the

circular queue's tail. RR approach is among the most widely used algorithms for time sharing in multiuser operating systems. [4, 5]. The size of the time interval affects how well the RR algorithm works (Each process is given a certain amount of time). If the chosen TQ is very large based on the RR technique, it will result in the starving problem (a process using a lot of CPU will be held for a long time) [6]. On the other hand, a short time interval will result in numerous context switches [7]. RR enhances response times and effectively utilises shared resources. Static time quantum is used, which results in longer waiting times, unwanted overhead and increased turnaround times for processes with different CPU bursts. To enhance the performance of RR algorithm, it automatically adapts to tasks in the queue by using a dynamic time quantum with RR. Although certain characteristics are not used by the algorithms currently in use for choosing a dynamic time quantum, these parameters have an impact on the scheduling process and the system performances [8].

The main contribution of this paper is to enhance the RR algorithm by selecting intelligent TQ for candidate processes in real time without affecting its fairness. It is suggested to use a new algorithm to dynamically change the time quantum at different ready queue states. The proposed algorithm outperforms the standard RR algorithm in terms of a variety of performance parameters, including average waiting time (AWT), turnaround time (ATT) and number of context switches according to a mathematical model developed to prove this. The experimental results show that the suggested modified RR algorithm performs better than the standard RR algorithm. By using a progressive time quantum that is repeatedly adjusted in accordance with the remaining burst time of active processes, the suggested technique resolves the issue. The processes are ordered in ascending order and implemented to each process to reduce turnaround time, waiting time and number of context switches. The disadvantages of the presented algorithms, such as Amended Dynamic Round Robin (ADRR) [9], Efficient Dynamic Round Robin (EDRR) [10], Hybrid Round Robin (HYRR)[11], Median Average Round Robin (MARR)[12] and Median Mean Round Robin (MMRR)[13] are that they provide a higher AWT and ATT. The contribution of this proposed algorithm is to: 1- Decreasing AWT. 2- Decreasing ATT. 3- Decreasing the number of context switches compared to RR, ADRR, EDRR, HYRR, MARR, and MMRR algorithms. The remainder of the paper is arranged as follows. In Section 2, the drawbacks of the RR scheduling algorithm are reviewed. Section 3 explains some related work. 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. Drawbacks of the standard RR scheduling algorithm

RR algorithm has many drawbacks, which are as the following:

2.1 Low throughput

Throughput is the number of processes completed per unit of time. RR has a large number of context switches as a result of its TQ, which lowers the overall system performance (throughput) [14].

2.2 High average turnaround time

Turnaround time is the total time it takes for the process to execute, from the time of submission until the time of completion [15] 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 processes' turnaround time, T_{cti} is the completion time of the processes; T_{ati} is the processes' arrival time, n is the processes' number and ATT is the processes' average turnaround time. The RR algorithm is distinguished by a high turnaround time.

2.3 High response time

Response time is the time between the submission of a process and the time at which it receives its first response (allocated to the CPU) [15]. RR algorithm is considered that has large response time.

2.4 High average waiting time

Waiting time is the total time the process spent in ready queue[16]. It is calculated as in Eq. (3).

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

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

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

Where WT_i is the processes' waiting time, T_{bti} is the processes' burst time and AWT is the processes' average waiting time. In RR, the process waits its turn to own the processor by waiting in the ready queue. Processes are forced to leave the processor and return to the waiting state because time quantum is present. The biggest drawback of this operation is the lengthy of the resulting average waiting time.

2.5 Context switching

The process must exit the CPU after the time slice is complete. The scheduler allots the CPU to the following process in the ready queue after storing the context of the current process in a stack or register[17]. Context switching, which wastes time and increases scheduler overhead [18].

Each scheduling algorithm has a unique set of properties that help determine which scheduling algorithm will be more useful for the current issue [19]. 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. Related work

Several CPU scheduling algorithms have been created for allocation processes. By combining the most advantages features of each algorithm to produce the best algorithm possible for the situation.

In [20], the author proposed the VORR (Variant On Round Robin) technique, one of the upgrades and additions to the RR algorithm. By establishing an effective TQ based on the median of burst times, it efficiently exploits the CPU. When compared to the standard RR algorithm and some of its improvements in terms of AWT, ATT and number of context switches, the experiments show good results. Additionally, it improves some RR algorithms' response times. In [21], The authors proposed the average max Round Robin algorithm. In this algorithm, processes are added to the ready queue (RQ) and scheduled for execution from there, indicating that they have already been added. Each process in the ready queue has a zero arrival time. The processes are arranged in ascending order, and for each process, the time quantum is equal to (average + maximum burst time)/2 is determined. As the first iteration of a process is finished, certain processes are executed, and then they are removed from the ready queue. The same procedure will be repeated until the ready queue is empty. The average wait time and turnaround time are then computed. In [22], a new median Round Robin algorithm has been presented (MMRRA). The authors used the square root of the process's median and highest burst time to calculate a dynamic TQ. An essential factor is included of this algorithm. The CPU will finish the processes if any process completes its first cycle of time quantum processing and its remaining burst time is larger than 20% of its total burst time; otherwise, the process will run for a second cycle of processing. In [23], a modified Round robin is suggested. It works on the concept of the dynamic time quantum. The dynamic time quantum is calculated considering priority and shortness as well as burst times of the processes. Using the shortness component, a new time quantum is determined for each round and each process. As a result, the algorithm includes components from the priority scheduling strategy and the Shortest Job First algorithm. The AWT and ATT are decreased. In [18], an Optimized Round Robin (ORR) algorithm is suggested for operating systems' time-sharing. The ORR and RR are experimentally compared. The experimental findings indicate that ORR outperforms at reducing AWT and ATAT. In [24], Priority-based Round Robin (PBRR) CPU scheduling technique is an improved Round Robin scheduler. It could be improved somewhat to be nearly RR. It considers priorities depending on task management. Each process is given a priority index, after which the processes in the ready queue are sorted by priority index. This approach chooses the first process in the ready queue, and the CPU is allotted for a time quantum interval. The allocated processes are moved to the back of the ready queue after the time period during which they were performed.

Once the processes have completed execution, they are removed from the ready queue and the AWT, ATT and response time are then computed. In [16], a multi-programmed operating system's RR algorithm is suggested. By dividing the ready queue into three smaller queues with the highest, medium, and lowest priorities, the authors improved the value of time quantum. A threshold value determines how much TQ is assigned to each of these sub-queues. For every sub-queue, this approach uses a distinct time quantum. Every process in every sub-queue

should have completed its execution. The results of this approach in terms of AWT and ATT have been decreased.

In [9], an amended dynamic Round Robin Scheduling algorithm (ADRR) is proposed. The authors used dynamic TQ. The lowest CPU burst time value is used to set the TQ. The authors determine a TQ threshold of 20 and then check a condition; set TQ to 20 if TQ is below the threshold (20). To prevent the value of TQ from getting too low and leading to a rise in the number of context switches, this condition is checked. TQ is adjusted after each cycle. Based on the CPU burst time, all tasks are ordered in the ready queue in ascending order. They are assigned to the CPU for a TQ. If a process's remaining CPU burst time less than half of the TQ, it will be pre-empted. Processes that were pre-empted are reinserted into the ready queue in ascending order. The same principle applies until all of the processes are finished. In [10], an Efficient Dynamic Round Robin algorithm (EDRR) is proposed by choosing a dynamic time quantum that would let a process to complete if the remaining execution time was less than or equal to 20% of the total execution time. The maximum burst time is founded from the available processes in the ready queue. The TQ is then calculated as a percentage of this time which is a 80 % of the maximal burst time. This algorithm improves the system's performance by reducing AWT and ATT. In [11], the authors implemented a novel scheduling technique to reduce the AWT, ATT, response time and number of context switches. It is called a hybrid Round Robin scheduling mechanism (HYRR). Dynamically calculating the time quantum using the mean and minimum of the burst time. The mean and lowest burst times are used in phase 1 to determine the enhanced time quantum (ETQ). Following the calculation, a high priority is granted to the process with the shortest burst time that is not already running in the CPU and is allocated 1 Enhanced quantum time in the CPU. Phase 1 is carried out up till a single CPU allocation is given to each process. In Phase 2 the processes are sorted in the ready queue in increasing 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. The current process is reallocated in the CPU if the burst time of the currently running process in the CPU is less than or equal to 1ETQ. The second phase is run until the ready queue is empty. In [12], The authors used a dynamic TQ to create a new RR algorithm. They used the median and average burst times for each process (MARR). The AWT and ATT are improved by this algorithm.

In [13], the author proposed a Median Mean Round Robin (MMRR) approach that improves the functionality of the RR algorithm. The proposed approach determines an ideal dynamic time quantum for each round based on the remaining burst time of the processes, which is generated as (median+ mean)/2. The experimental data show that the performance has improved in terms of waiting time, turnaround time and context switching.

All of the enhancements of the RR CPU scheduling algorithms that are discussed above, they have certain drawbacks. The processes that enter the system may have varied burst times, which mean that their CPU execution times may also vary. Time quantum can either be high or low according to the RR algorithm. So this paper proposes a novel algorithm which has a dynamic optimal TQ and taking the remaining burst time of the processes into account to solve this problem by enhancing the performance of the system by: maximizing CPU utilization, minimizing waiting time and turnaround time and reducing number of context switches[25].

4. Proposed algorithm (Modified Median Mean Round Robin Algorithm (MMMRR))

Because of the drawbacks of the RR approach, many improvements have developed but still have some issues. Therefore, in order to maximize the act of a scheduling algorithm, the proposed approach has focused mostly on determining an effective time quantum. The proposed method first sets the ready queue in an increasing order based on the burst time of the processes, and then establishes the time quantum based on the mean and the median of execution times. If every process enters the ready queue simultaneously, they are ordered according to their burst time in increasing order. Then the TQ 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 all processes' burst time 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 process' burst time, n is the processes' number. At beginning, the CPU is assigned to the first process in the ready queue. The burst time of the currently active process is then checked for the remaining CPU. If the CPU burst time is less than the TQ, the CPU is once more reallocated to the active process for the remaining time. Otherwise, the process will be terminated to the tail of the ready queue. After each process is allocated, if the ready queue is empty and all processes are completed its execution, then the average of waiting time and turnaround time is calculated. Hence, the proposed algorithm enhances the performance of the system. The flowchart of the proposed algorithm is described as follow in Fig. 1.

5. Experimental results

In order to demonstrate the efficiency of our technique and to ensure a fair comparison between the proposed algorithm and the RR, ADRR[9], EDRR[10], HYRR[11], MARR[12] and MMRR[13] algorithms, we select cases that are used in the majority of these algorithms and have the same number of processes, burst time, and arrival time. There are the datasets which consist of 15 processes that have been considered.

Modified Median Mean Round Robin algorithm:

1. Assign processes into the ready queue.
2. all the processes are sorted in an increasing order depending on their burst time
3. TQ ← ⌊(median + mean) / 2⌋
4. While (ready queue! =NULL)
5. If (remaining burst time < TQ)
6. The CPU is allocated again to the current running process for the remaining burst time.

   Else

7. Add the remaining of current process to the end of the ready queue.
8. Go to step 4
9. End while
10. Calculate AWT and ATT.

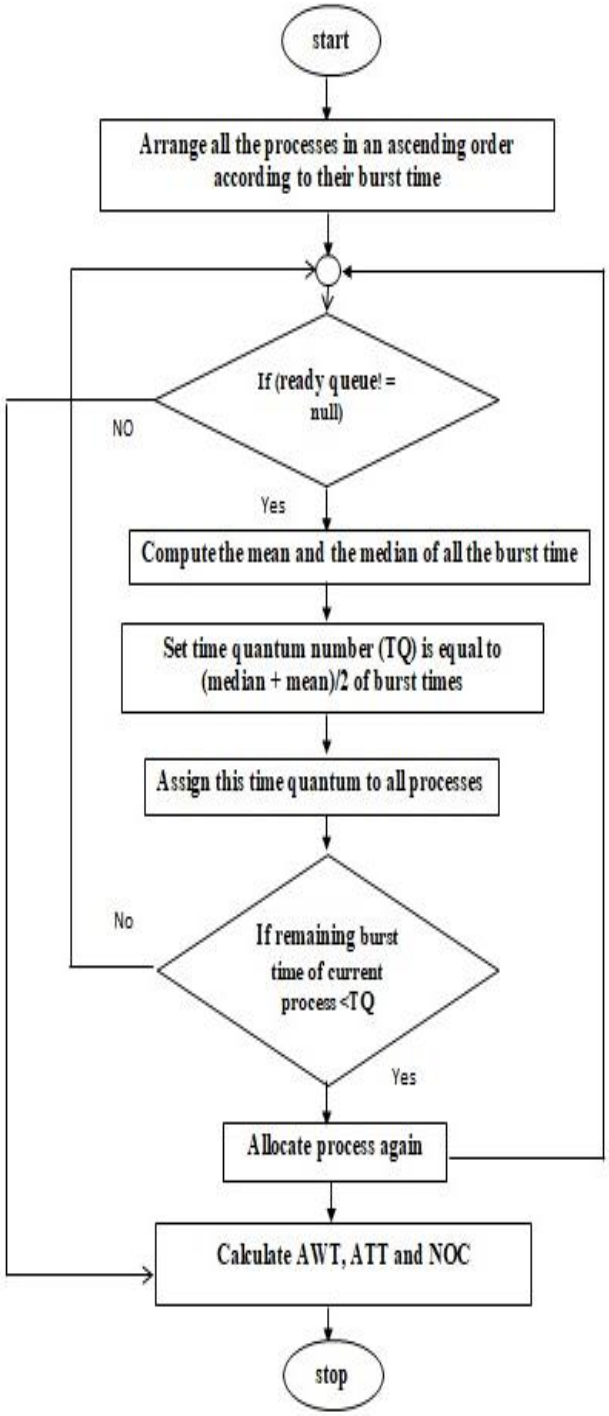

Figure. 1 Flowchart of the MMMRR algorithm

5.1 Case 1: Processes are in random

The processes are arriving at zero time with random burst time as shown in Table 1. Table 2 presents a comparative study among the RR, ADRR[9], EDRR[10], HYRR[11], MARR[12] and MMRR[13] with respect to TQ, AWT and ATT for case 1.

The comparison of AWT and ATT for the current algorithms is shown in Fig. 2 below.

Table 1. Processes are coming in random, increasing and decreasing order

Case 1: random order		Case 2: increasing order		Case 3: decreasing order	
Process	**Burst time**	**Process**	**Burst time**	**Process**	**Burst time**
P1	42	P1	35	P1	250
P2	68	P2	40	P2	186
P3	135	P3	55	P3	174
P4	101	P4	60	P4	163
P5	170	P5	75	P5	146
P6	125	P6	80	P6	140
P7	79	P7	94	P7	132
P8	159	P8	101	P8	114
P9	163	P9	112	P9	100
P10	65	P10	121	P10	97
P11	106	P11	125	P11	88
P12	146	P12	134	P12	37
P13	82	P13	140	P13	20
P14	28	P14	180	P14	18
P15	162	P15	197	P15	12

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

Algorithm	TQ	AWT (ms)	ATT (ms)
RR	35	1091	1199.73
ADRR	28,20,20, 20,20,20, 20,20	1012.33	1121.06
HYRR	69	820.8	929.53
EDRR	136,170	645.8	754.53
MARR	108,48,7, 7	738.2	846.93
MMRR	107,57	589.2	697.93
proposed	107	567.8	676.53

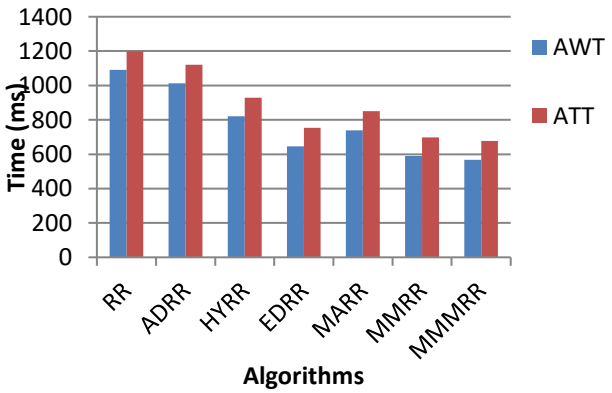

Figure. 2 Comparative graph for the AWT and ATT (case 1)

5.2 Case 2: Process are coming in increasing order

The processes are arriving at zero time with increasing burst time as shown in Table 1. A comparison of the RR, ADRR[9], EDRR[10], HYRR[11], MARR[12] and MMRR[13] algorithms with respect to TQ, AWT and ATT for case 2 is shown in Table 3 below.

The comparison of AWT and ATT for the current algorithms is shown in Fig. 3 below.

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

5.3 Case 3: Process are coming in decreasing order

The processes are arriving at zero time with decreasing burst time as shown in Table 1.

A comparison of the RR, ADRR[9], EDRR[10], HYRR[11], MARR[12] and MMRR[13] algorithms with respect to TQ, AWT and ATT for case 3 is shown in Table 4 below.

Table 3. Comparative study of RR, ADRR, HYRR, EDRR, MARR, MMRR and proposed algorithm (case 2)

Algorithm	TQ	AWT (ms)	ATT (ms)
RR	35	920.86	1024.13
ADRR	35,20,20, 20,20,20, 45	872.53	975.8
HYRR	70	783.2	886.46
EDRR	157,197	526.53	629.8
MARR	103,37,49 ,8	673.2	776.4
MMRR	102,86	533.33	636.6
proposed	102	526.53	629.8

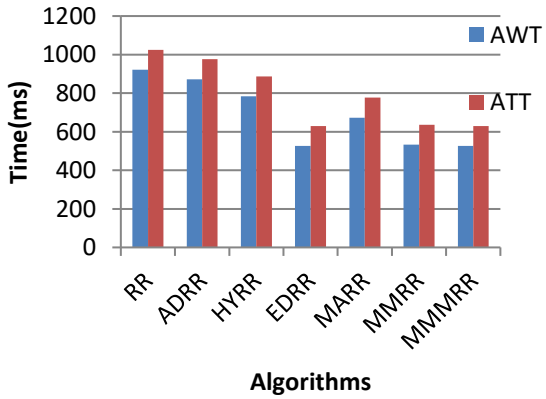

Figure. 3 Comparative graph for the AWT and ATT (case 2)

Table 4. Comparative study of RR, ADRR, HYRR, EDRR, MARR, MMRR and proposed algorithm (case 3)

Algorithm	TQ	AWT (ms)	ATT (ms)
RR	35	1100.13	1211.93
ADRR	20,20,48,26 ,20,20,20,2 0,56	805.93	917.73
HYRR	62	727.26	839.06
EDRR	200,250	927.06	1038.86
MARR	113,46,28,6 3	729.26	841.06
MMRR	112,82,56	522.33	634.13
proposed	112,138	499.93	611.73

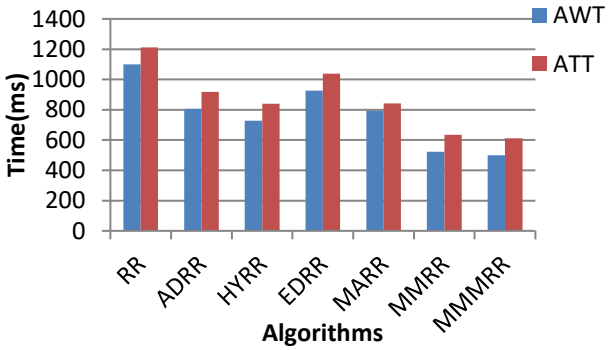

Figure.4 Comparative graph for the AWT and ATT (case 3)

The comparison of AWT and ATT for the current algorithms is shown in Fig. 4.

5 Result analysis

Proposed algorithm (MMMRR) is compared with Amended Dynamic Round Robin (ADRR) [9], Hybrid Round Robin (HYRR) [11], An efficient Dynamic Round Robin (EDRR) [10], Median average Round Robin (MARR) [12] and Median Mean Round Robin (MMRR) [13]. These algorithms are all compared with the RR algorithm in order to evaluate their performance. Proposed algorithm (MMMRR), RR, and other algorithms are implemented in C ++ and compared using the same random data set. The comparison of these algorithms is based on AWT and ATT. Because of the number of processes in the ready queue determines AWT and ATT, 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. Comparing algorithms based on their average waiting times is shown in Fig. 5 and Fig 7. For (10 to 100) and (500 to 5000) processes, in the ready queue, the stacked line graph is plotted. The number of processes in the ready queue is plotted against the x-axis, and the average waiting time for the processes is provided in milliseconds and plotted by the y-axis. The proposed algorithm

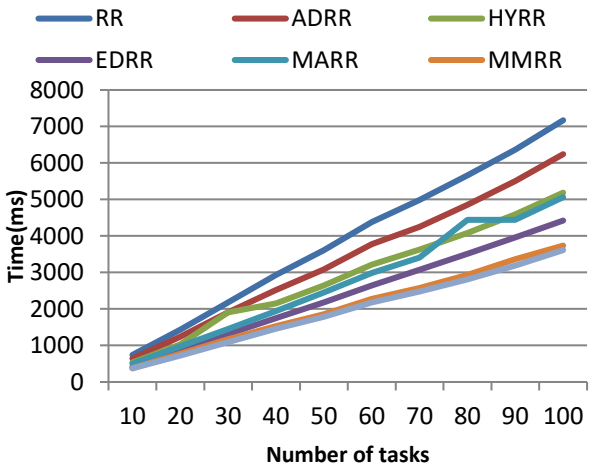

Figure. 5 Comparative graph for the average waiting time

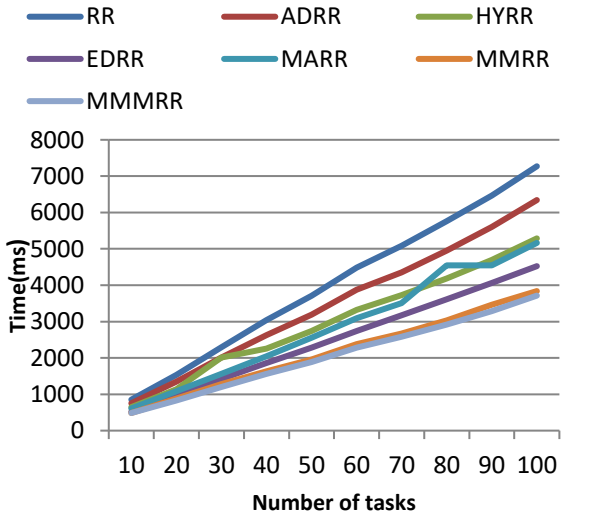

Figure. 6 Comparative graph for the average turnaround time

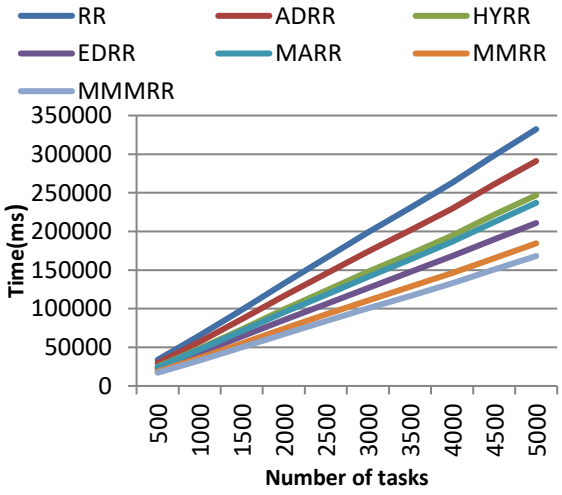

Figure. 7 Comparative graph for the average waiting time

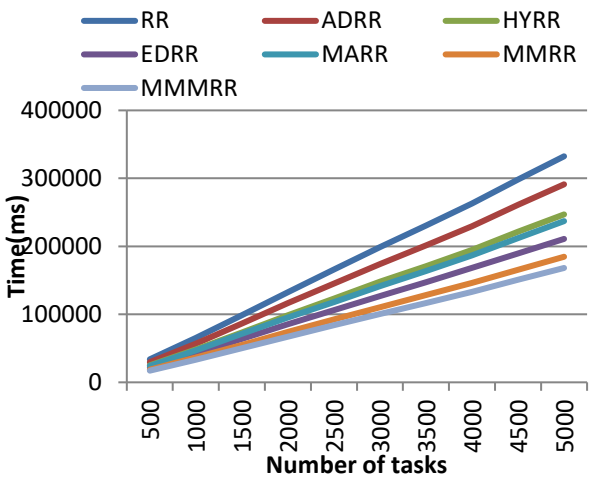

Figure. 8 Comparative graph for the average turnaround time

(MMMRR) gives better results, followed by MMRR [13], EDRR [10], MARR [12], HYRR [11] and ADRR [9]. A substantial improvement is given by these algorithms when compared to RR algorithm. With an increase in processes, the performance of the algorithms is enhanced. The MMRR, 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 with an increase in processes, the performance of MMMRR showed an upward trend in average waiting time compared to other algorithms. In comparison to suggested algorithms, the AWT 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 6 and Fig 8. For (10 to 100) and (500 to 5000) processes, in the ready queue, the stacked line graph is plotted. The number of processes in the ready queue is plotted by the x-axis, and the average turnaround time for the processes is provided in milliseconds and plotted by the y-axis. The proposed algorithm (MMMRR) gives better results, followed by MMRR[13], EDRR [10], MARR [12], HYRR [11] and ADRR [9]. A substantial improvement is given by these algorithms when compared to RR algorithm. As the number of processes in the ready queue rises the performance of the algorithms is enhanced. The MMRR, 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 algorithm act similarly, however with an increase in processes, the performance of MMMRR 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.

6 Conclusion and future work

Many researchers have been developed the RR algorithm for task scheduling to appropriate to any system. The TQ is the most crucial problem with the RR method. As opposed to the static TQ that is used in the case of the traditional RR method, the MMMRR operates on the concept of dynamic TQ. Each process in a static TQ system is given a certain time slice during which the CPU will carry out the assigned task. Due to its dynamic design, the time-quantum for each cycle of CPU allocation for each task in the ready queue is varying. The main contribution of this research is the suggestion of a Modified Median Mean Round Robin algorithm (MMMRR), which improves the performance of the RR method. It established a dynamic time quantum that is computed as (median + mean)/2 while also taking into account the remaining burst time of the active process.

If the current process's remaining burst time is less than the time quantum, it will allocate again. If not, it gets moved to the end of the ready queue. Based on the burst time for these processes, each cycle's TQ will be determined. Utilizing a variable TQ based on burst time allowed for the reduction of AWT, ATT and number of context switches. The experimental results showed that the proposed algorithm improves system performance by lowering the AWT, ATT and number of context switches. The proposed MMMRR algorithm successfully optimised the AWT, ATT and number of context switches when compared to the RR, ADRR, HYRR, EDRR, MARR and MMRR algorithms. In the future work, we will improve the RR algorithm by developing an algorithm of divided sub queues with different calculations values of time quantum.

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] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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 (DATABIA)*, pp. 63-68, 2020.

[6] 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 (ICCAIS).IEEE*, pp. 1-7, 2019.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Sakshi, C. Sharma, S. Sharma, S. Kautish, S. Alsallami, E. Khalil, 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.

[13] N. Ghazy, A. Abdelkader, M. Zaki, and K. ElDahshan, "A New Round Robin Algorithm for Task Scheduling in Real-time System", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 5, pp. 691-704, 2022, doi: 10.22266/ijies2022.1031.59.

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

[15] K. ElDahshan, 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.

[16] 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.

[17] 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.

[18] 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.
[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] A. Abdelhafiz, "VORR: A NEW ROUND ROBIN SCHEDULING ALGORITHM", *Al-Azhar Bulletin of Science : Section B*, Vol. 32, No. 1-B, pp. 27-44, 2021.
[21] 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 (IJITEE)*, No. 1, pp. 2278-3075, 2012.
[22] H. Mora, S. Abdullahi, and S. Junaidu, "Modified Median Round Robin Algorithm (MMRRA)", In: *Proc. of 13th International Conference on Electronics, Computer and Computation (ICECCO). IEEE*, pp. 1-7, 2017.
[23] R. Kumar, "Modified Round Robin Algorithm based on Priority and Shortness components", Vol. 9, No. 5, pp. 72-76, 2021.
[24] S. Zouaoui, L. Boussaid, and A. Mtibaa, "Priority based round robin (PBRR) CPU scheduling algorithm", *International Journal of Electrical & Computer Engineering*, Vol. 9, No. 1, pp. 2088-8708, 2019.
[25] 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.