An Improvement of Round Robin Scheduling Algorithm

Md. Shafiul Alam Forhad[1], Mrinmoy Das[2], Md. Monowar Hossain[3]
[1]Department of Computer Science & Engineering, Chittagong University of Engineering & Technology, Bangladesh
[2]Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Bangladesh
[3]Department of Computer Science & Engineering, Bangabandhu Sheikh Mujibur Rahman Science and Technology University ,Gopalganj, Bangladesh

Summary
The processor is one among the foremost imperative parts of computer system. CPU scheduling regulates foremost powerful way to serve the pending appeals of the processes. The maximum algorithm of CPU scheduling emphasis on supreme CPU usage and diminishing waiting time, turnaround time and throughput for a gang of requests. In the time measured scheme, the Round Robin algorithm is the appropriate selection. It is inappropriate for actual time schemes because of higher WT(Waiting Time), higher TT(Turnaround Time) and a large amount of NCS(No. of Context Switches. Selection of TQ (Time Quantum) is incredibly vital. Time quantum influences algorithm's performance. The effectiveness of the algorithm totally relies on the selection of time quantum. Here, an improved version of the Round Robin(RR) algorithm, FMMRR (Forhad Mrinmoy Monowar Round Robin) has been proposed. It needs dynamic TQ rather than static TQ needed in RR. The behavior of our recommended FMMRR algorithm is experimentally distinguished with RR and a few other improved versions of RR. The experimental output of our method demonstrates improved performance in terms of some scheduling criteria such as average TT, average WT, throughput etc. Once enhancement in RR, it's been initiated that WT and TT have been shortened remarkably.

Key words:
Average Turnaround Time, RR Scheduling, Operating System, Context Switching, Average Waiting Time.

1. Introduction

An operating system is an enlarged machine from user's perspective and a resource manager from system's perspective. It is a software that supports the basic functions of a computer, scheduling tasks, etc. to abet programmers in improving system competence and stability. In multiprogramming and multitasking environments, it is essential to accept the process among the number of processes exist in the job pool according to their demand. Distribution of CPU to the processes is completed by the scheduler, which is conducted by some scheduling algorithms such as FCFS, SJF, Priority & RR.
In which RR is the most popular preemptive scheduling algorithm. In non-preemption, CPU is allowed to do a task until its processing is finished. On the other hand, the functioning task is compelled to free CPU by the recently appeared process in preemption. Every scheduling algorithm has its own benefits and drawbacks. Similarly, RR has a drawback which increases average turnaround time, waiting time and minimizes throughput known as a context switch. The processes in RR are assigned with a TQ which is static by nature.
In RR scheduling, processes acquire an equitable share of CPU because of static time quantum allocated to every process and also context switch is inversely proportional to the selection of static TQ and it degrades the overall performance of the system. All performance includes high average TT and WT. This kind of factor provokes us to plan a refined algorithm. By diminishing the number of context switches, average TT and WT utilizing perception of dynamic time TQ, it is qualified to enhance the system execution. In our task, we improved the RR algorithm by utilizing dynamic TQ wisely and we also use sorting for the sequencing of processes. This approach drastically diminishes average WT (AWT), average TT(ATT) and context switching.

2. Background

RR algorithm acts on time allocating aspect. A time portion is appointed to all processes. Each process will be beheaded for specific time portion. In the RQ(Ready Queue), the processes which are ready for execution is placed. If any process arrives, then they are added in the tail of the RQ. From that the processes are selected by the CPU scheduler and then fix the timer to a specific time period. If that process still unfinished its entire execution within a specific time period, then it will be occupied after completion of TQ and included at the tail of RQ [1]. Then the upcoming process in the front of the RQ is assigned to the CPU. All the process completes their execution in this manner.
RR CPU scheduling algorithm is given below:
Step-1: START
Step-2: Build a RQ of the processes like READYQ.
Step-3. Take the no. of processes, execution time and other parameters as inputs.
Step-4: Ready processes will put into READYQ
Step-5: Then REPEAT step no 6, 7 and 8 respectively till READYQ becomes empty.
Step-6: Choose the first process from READYQ and assign CPU for the given TQ for executing that process.

Manuscript received December 5, 2018
Manuscript revised December 20, 2018

Step-7: After completing the execution of the ongoing process, took away it from READYQ and move to step no 5 again.
Step-8: Then eliminate the ongoing executing process from the READYQ and set down it at the tail of READYQ.
Step-9: Calculate AWT, ATT, and NCS.
Step-10: Exit

3. Related works

In [2], they arrange the processes as specified by their burst time and allocating them with optimal TQ. It is able to reduce the drawbacks of the existing algorithms by taking dynamic TQ. The ATT, AWT & NCS are greatly shortened by this idea.
In [3], they proposed an approach for RR scheduling. It helps to enhance the effectiveness of CPU in both real-time and time-sharing OS (Operating System). A comparative study of the presented methodology with traditional one is shown. The authors feel that the proposed algorithm takes care of all the issue experienced in simple RR algorithm by diminishing the performance parameters. The drawbacks of traditional RR algorithm eliminated by their proposed algorithm and also increasing the system throughput.
In [4], an improved version of the RR scheduling algorithm, IRRVQ is proposed. The outputs of this algorithm show that TT(Turnaround Time) and WT(Waiting Time) have been decreased. The presented algorithm is experimentally justified better than typical RR and giving better performance than conventional RR algorithm.
In [5], they proposed a dynamic TQ-based algorithm of RR scheduling to improve the functioning of CPU and the results display that presented algorithm is more suitable than RR, IRR & IRRVQ in respect of different parameters. And that parameters are the AWT, NCS, and ATT.
In [6], they proposed an algorithm called Revamped Mean Round Robin (RMRR). Their algorithm is based on improvement and revised on the simple RR algorithm and it was compared with other algorithms. RMRR together with the RR algorithm was implemented in python and results show that the proposed technique has minimal AWT, ATT and therefore the Numbers of Context Switches.
In [7], DABRR (Dynamic Average Burst Round Robin) has been proposed which utilizes dynamic TQ within the place of static TQ needed in Round Robin. The DABRR algorithm's performance is contrasted with RR and a few other prevailing variants of Round Robin. They show DABRR improves performance in respect of NCS, AWT, and ATT.
In [8], a newly suggested variant of this algorithm submitted, analyzed in details, checked and proved. The newly suggested algorithm known as Self-Adjustment-RR (SARR) in view of dynamic TQ. This method's concept is to create TQ frequently adapted as stated by the execution time of current working processes in view of the research and computations they have constructed. New moderated algorithm profoundly finds an answer to the problem of fixed TQ that is treated as a threat to the RR algorithm. Dynamic scheduling algorithm implementation expanded execution & steadiness of OS and comfort making of an OS that is self-adjusted.
In [9], CPU scheduling algorithm with increased performance has been proposed. Pipelining method is employed for speed up issue. That method may be applied to some algorithms of CPU scheduling to enhance execution. The testing demonstrates that the suggested algorithm is better than the existing planning algorithms. The performance is increased by 40-50%.
Babu.B et al. proposed an efficient RR algorithm[10] for the multi-programmed OS. With regard to some usual scheduling algorithms like First Come First Served, Round Robin etc., they initiated a tool which delivers output according to experimental results.
In [11], they have developed an RR algorithm.Their algorithm is based on dynamic time slice. They tried to lessen running time of an algorithm and also minimizing the NCS, AWT and ATT.
In [12], they have implemented a genetic approach based RR algorithm. Static, Dynamic TQ and Genetic algorithm-based RR with dynamic time quantum are compared with the proposed method. The quantum can be dynamically selected for all iterations. Instead of keeping it static, it can also enhance the performance.
In [13], they proposed an algorithm that categorizes the processes as high and low priority processes. The proposed method decreases the AWT time of high priority process regardless of the low priority process. The overall AWT will adjust according to the considered set of processes. Based on the AWT, it is justified that the proposed scheme gives shortened AWT of the process set than previously proposed schemes.
In [14], they proposed an improvement of RR CPU scheduling algorithm. This proposed algorithm reduces the AWT, ATT and the number of context switches. The RR algorithm with non-preemptive SJF method is applied for getting better performance.
In [15], OMDRRS algorithm is developed. Better improvement is seen in WT, TT and context switching. This algorithm provides less WT, less TT and context switching. It leads to save plenty of memory space and thus reducing the overhead. It boosts the performance of traditional RR and some other algorithms.
Parashar, Mayank, and Amit Chugh. proposed an algorithm [16] which is compared against the outputs of

the existing RR algorithm. By modifying the value of the TQ for some processes, it boosts it further. In this case, they only consider the above TQ for the processes that need a little greater TQ than the assigned TQ cycles. During this proposed approach, it doesn't aim to alter the philosophy of the conventional RR algorithm. However, within the conventional RR manner, the remaining processes are executed

In [17], they proposed a technique which improves the OS scheduling. It improves, particularly within the time measuring system. By using the geometric mean, the value of the TQ is calculated. Their proposed technique applies the Shortest Job First algorithm and adopts a proper process to be scheduled and executed. They wanted to execute the presented method on TT & response time. That proposed method will enhance the TT, response time in their upcoming work.

In [18], a new RR scheduling, IMRRSJF proposed. If no. of process is high, then IMRRSJF gives a better result. It offers a better result compared to RR, ERR, IRR etc.

In [19], they proposed a new scheduling algorithm better than the variants of RR and a further Improvement RR(AAIRR) algorithms in terms of minimizing the NCS, AWT and ATT.

4. The Proposed Method

In this section, our proposed FMMRR(Forhad Mrinmoy Monowar Round Robin) algorithm are described in details.

4.1 FMMRR Algorithm

TQ: Time Quantum
RQ: Ready Queue
n: No. of processes in the RQ
Pi: Process at ith index
i, j: Used as an index of the ready queue
AT: Arrival Time
WT: Waiting Time
TT: Turnaround Time
BT: Burst Time
TBT: Total BT
AWT: Average WT
ATT: Average TT
[1] Start
[2] Sort the processes in RQ according to their BT in ascending order
[3] i=1, TBT=0
[4] Repeat steps 5 and 6 till i <= n
[5] TBT += burst time of process Pi
[6] i++
[7] TQ = Time Quantum
[8] Calculate the median of BT of all RQ processes
[9] If the number of process less than or equals to three
[10] Then TQ= Largest CPU burst among all processes
[11] Otherwise TQ= Median
[12] j = 0
[13] Repeat from step 14 to 26 till j<n
[14] If (burst time of Pi) <= TQ
[15] Then execute the process
[16] Take the process out of the ready queue
[17] n--
[18] Else
[19] Execute Pi for a time period up to 1 TQ
[20] BT of Pi = BT of Pi – TQ
[21] Add the process to the RQ for next round of execution
[22] j++
[23] If a new process arrives
[24] goto [2]
[25] If RQ is not empty
[26] goto [3]
[27] Calculate the number of context switches
[28] Calculate AWT
[29] Calculate ATAT
[30] End

Forhad Mrinmoy Monowar Round Robin (FMMRR) algorithm is preemptive in nature. At first, the proposed method takes the no. of processes, their arrival time and also their CPU execution time as inputs. Then the processes which are in the RQ and waiting for execution are organized ascending order of their CPU execution time. After that, total burst time and median are calculated. The processes are executed according to a TQ, which is allocated to each process. If the TQ of a process expires before completion, then the next incoming process is assigned and before that at the tail of the RQ, the currently running process is placed. The first process from the RQ is picked by the CPU scheduler. If a new process arrives, then RQ processes are again sorted. If the no. of processes less than or equals to three, then the value of the TQ is the largest CPU burst time among all the remaining processes in the RQ. In other respects, the value of the TQ is set to the median of CPU burst among all the remaining processes. Then the process executes if the remaining BT of the process is less than or equals to the TQ and after execution, it removes from the RQ. The number of RQ processes decreased. Otherwise, the process executed for a time interval. Then include it to the end of the RQ. This process repeats until RQ is not empty.

5. Experimental Findings

This section provides the comparative analysis of RR, IRRVQ, DABRR, SARR and Forhad Mrinmoy Monowar Round Robin algorithms on the basis of their resulted context switches, WT and TT.

Let us consider five processes Pr1, Pr2, Pr3, Pr4 and Pr5 respectively. Their AT, execution time is shown in Table 1. Gantt chart for this set of processes is shown in Figure 2. The burst time median of all processes is 55. So, this value is chosen as the value of the TQ. After 1st iteration, the number of currently remaining processes in the RQ is less than 3. The remaining highest BT of the RQ processes is chosen as the new value of the TQ. So, 50 is the new value of the TQ. And if the number of remaining processes in the RQ is greater than three, then again the median of BT of the remaining processes is chosen as TQ.

Table 1: Processes with Zero AT and BT in decreasing order [7]

Processes	AT	BT
Pr_1	0	105
Pr_2	0	85
Pr_3	0	55
Pr_4	0	43
Pr_5	0	35

55					50	
Pr_5	Pr_4	Pr_3	Pr_2	Pr_1	Pr_2	Pr_1
0 35	78	133	188	243	273	323

Fig. 1 Gantt chart of each RR algorithm for Table 1

Again, consider five processes Pr1, Pr2, Pr3, Pr4 and Pr5 respectively. Their AT and CPU BT are shown in Table 2. Gantt chart for this set of processes is shown in Figure 5. The median value is calculated and it is 75. So 75 is chosen as the new value of the TQ. After 1st iteration, the number of currently remaining processes in the RQ is less than 3. A new value of the TQ is chosen from the remaining highest processes in the RQ. So, 60 is the new value of TQ.

For illustration, let us again consider five processes Pr1, Pr2, Pr3, Pr4, and Pr5 respectively. Their arrival time and CPU BT is shown in Table 3. Gantt chart for this set of processes is shown in Figure 9. As there is no process except Pr1 in the RQ queue at zero AT. So the TQ is set to the value of the largest TQ and it is 45. Then after 1st iteration, the value BT of the processes in the RQ. The value is 62. Then after the 2nd iteration, only two process exists in the RQ. So, the TQ is the largest BT of the remaining processes and it is 28.

Figure 2, 3 and 4 shows the graphical analysis of the context switch, WT and TT respectively of RR, IRRVQ[4], DABRR[7], SARR[8] and Forhad Mrinmoy Monowar Round Robin algorithms for the set of processes shown in Table 1. By evaluating these three graphs we conclude that Forhad Mrinmoy Monowar Round Robin algorithm executes far better than four other algorithms in case of processes arrived at zero AT with descending order of BT.

Figure 6, 7 and 8 shows the graphical analysis of the context switch, WT and TT respectively of RR, IRRVQ[4], DABRR[7], SARR[8], and Forhad Mrinmoy Monowar Round Robin algorithms for the set of processes shown in Table 2. By evaluating these three graphs we conclude that Forhad Mrinmoy Monowar Round Robin algorithm executes far better than the four other algorithms in case of processes arrived at zero arrival time with random BT.

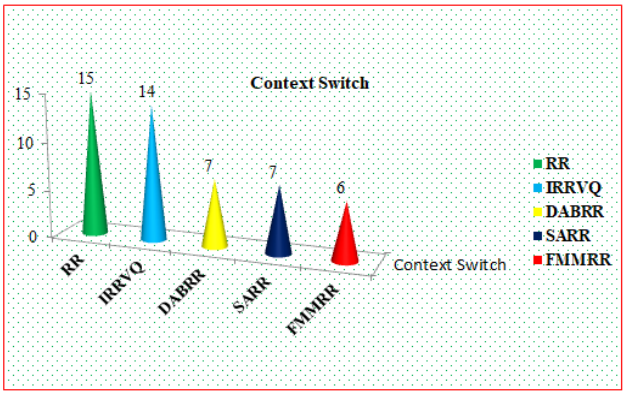

Fig. 2 Context switch of each RR algorithm for Table 1

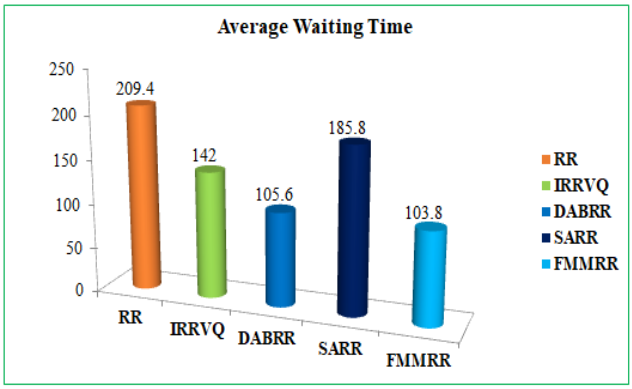

Fig. 3 AWT of each RR algorithm for Table 1

Figure 10, 11 and 12 shows the graphical analysis of the context switch, WT and TT respectively of RR, IRRVQ[4], DABRR[7], SARR[8], and Forhad Mrinmoy Monowar Round Robin algorithms for the set of processes shown in Table 3. By evaluating these three graphs we conclude that Forhad Mrinmoy Monowar Round Robin algorithm executes far better than four other algorithms in case of processes arrived with different arrival times with random BT.

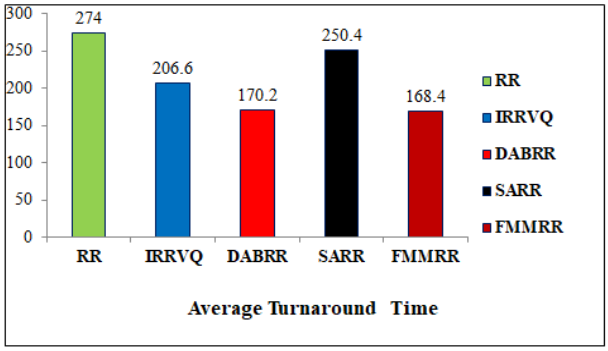

Fig. 4 ATT of each RR algorithm for Table 1

Table 2: Processes with Zero AT and BT in random order [7]

Processes	AT	BT
Pr1	0	105
Pr2	0	60
Pr3	0	120
Pr4	0	48
Pr5	0	75

75					60	
Pr_4	Pr_2	Pr_5	Pr_1	Pr_3	Pr_1	Pr_3
0 48	108	183	258	333	363	408

Fig. 5 Gantt chart of each RR algorithm for Table 2

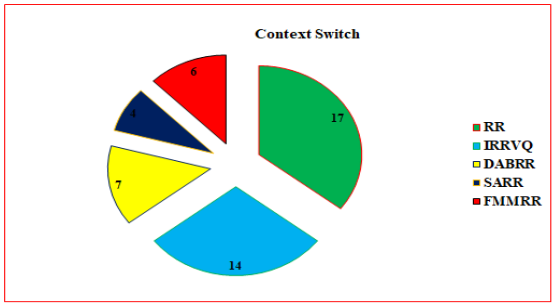

Fig. 6 Context switch of each RR algorithm for Table 2

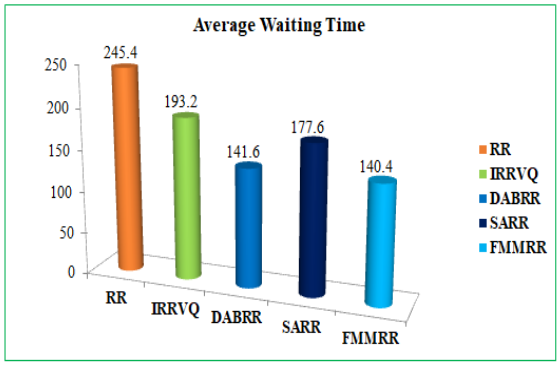

Fig. 7 AWT of each RR algorithm for Table 2

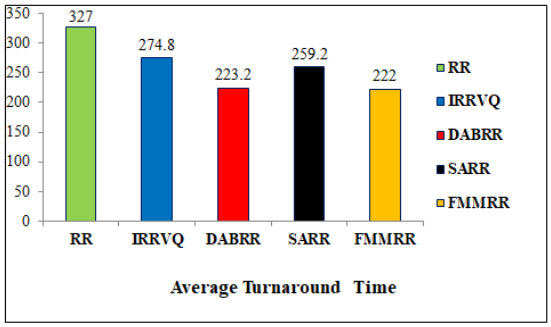

Fig. 8 ATT of each RR algorithm for Table 2

Table 3: Processes without Zero AT and BT in random order. [7]

Processes	AT	BT
Pr1	0	45
Pr2	5	90
Pr3	8	70
Pr4	15	38
Pr5	20	55

45	62				28	
Pr_1	Pr_4	Pr_5	Pr_3	Pr_2	Pr_3	Pr_2
0 45	83	138	200	262	270	298

Fig. 9 Gantt chart of each RR algorithm for Table 3

6. Results and Discussion

Forhad-Mrinmoy-Monowar-Round-Robin algorithm was developed to resolve all earlier revealed problems by using dynamic-time-quantum in a simple, feasible and significant

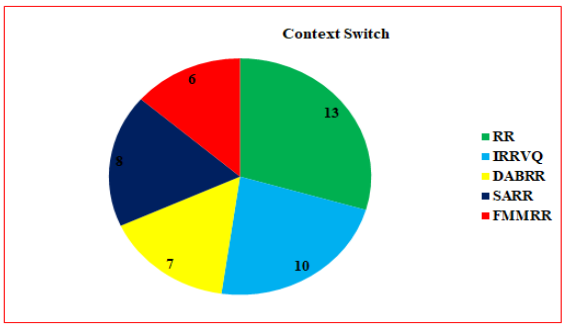

Fig. 10 Context switch of each RR algorithm for Table 3

manner. The contrasts shown in the above figures proved that the presented algorithm FMMRR gives much better results than other in some scheduling criteria. We can conclude from the above experimental results that the presented FMMRR algorithm functions far better than RR, IRRVQ[4], DABRR[7] and SARR[8] in respect of AWT and ATT. In Fig.13, the output of the FMMRR algorithm for descending order of burst time with zero AT is shown. The pictorial representation of the execution of the

processes in the CPU using Gantt chart, AWT and ATT is shown in the Fig.13. In the Fig. 14, the output is shown for random order of burst time with zero AT is shown. The result shows that the algorithm works good and satisfactorily for a large number of processes and is able to give better AWT and ATT for the majority of processes.

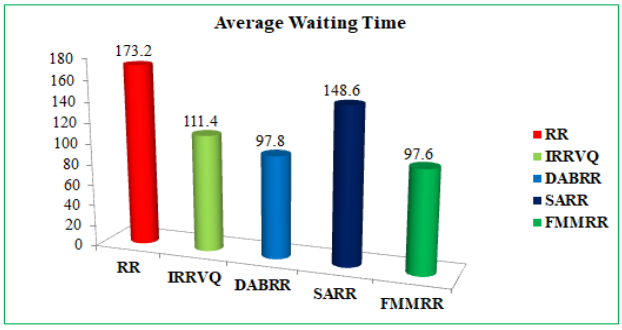

Fig. 11 AWT of each RR algorithm for Table 3

7. Conclusion

In this paper, Forhad Mrinmoy Monowar Round Robin algorithm, which is an improved version of the RR

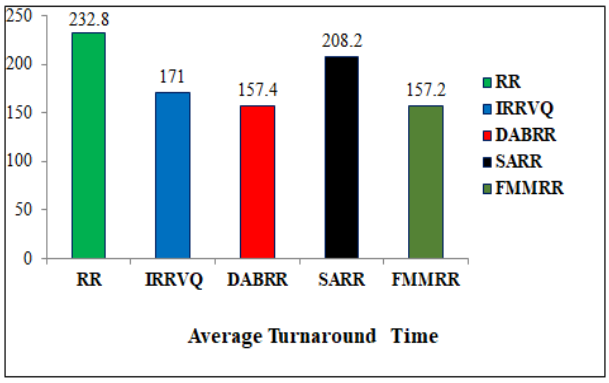

Fig. 12 ATT of each RR algorithm for Table 3

algorithm is proposed. There are some changes here in this algorithm like the value of the TQ. Forhad Mrinmoy Monowar Round Robin provides better performance than the mentioned variants of RR algorithms. This can be achieved by minimizing the AWT, ATT, etc. The AWT, ATT has been computed and then the results were compared and have appeared that the presented algorithm

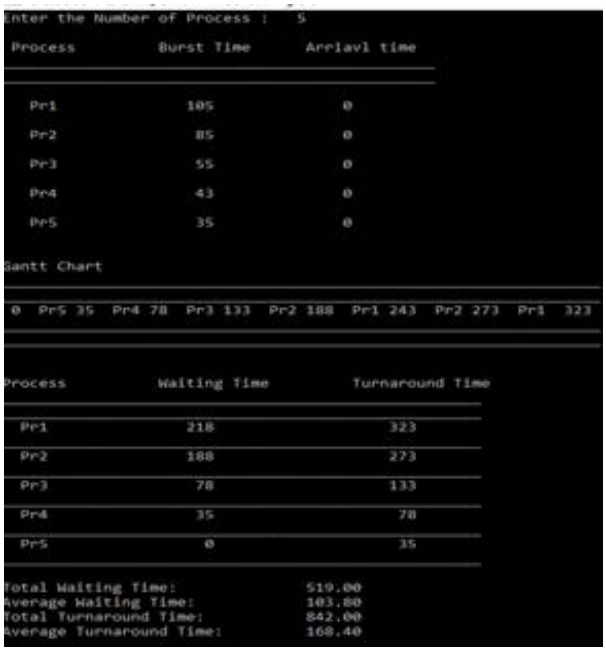

Fig. 13 Experimental results of FMMRR algorithm for Table 1

offers far better results than RR, DABRR, IRRVQ, SARR in respect of above parameters. It's quite easy to implement in a practical situation. In the future, we want to improve a better version of this algorithm.

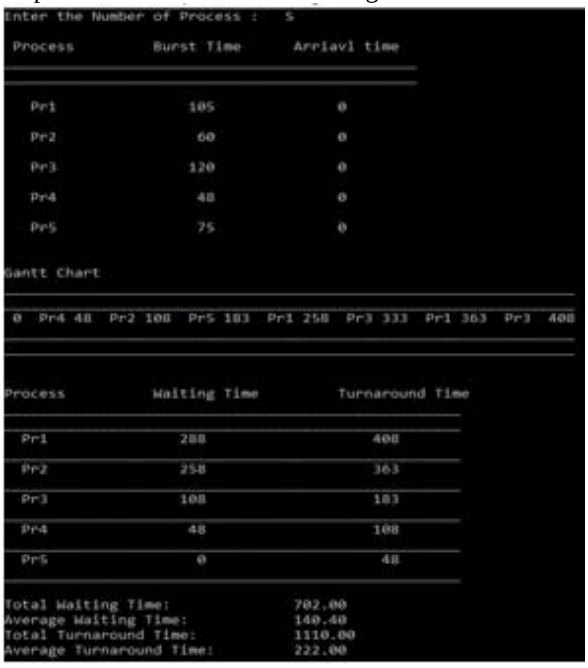

Fig. 14 Experimental results of FMMRR algorithm for Table 2

References

[1] Amit, Narendra and Dinesh Goyal. "Review of Round Robin (RR) CPU Scheduling Algorithm on Varying Time

Quantum." International Journal of Engineering Science Invention, vol. 6, no. 8, August 2017, pp 68-72.
[2] Nayak, Debashree, et al. "Improved Round Robin Scheduling using Dynamic Time Quantum." International Journal of Computer Applications, vol. 38, no. 5, 2012, pp. 34-38.
[3] Singh, Ajit, et al. "An Optimized Round Robin Scheduling Algorithm for CPU Scheduling." (IJCSE) International Journal on Computer Science and Engineering, vol. 02, 7 Nov. 2010.
[4] Kumar Mishra, Manish, and Faizur Rashid. "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum." International Journal of Computer Science, Engineering and Applications, vol. 4, no. 4, 2014, pp. 1-8.
[5] Berhanu, Yosef. "Dynamic Time Quantum based Round Robin CPU Scheduling Algorithm." International Journal of Computer Applications, vol. 167, no. 13, 2017, pp. 48-55.
[6] Kathuria, Sachin, et al. "A Revamped Mean Round Robin (RMRR) CPU Scheduling Algorithm." International Journal of Innovative Research in Computer and Communication Engineering, vol. 4, no. 4, Apr. 2016.
[7] Dash, Amar R., et al. "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum." International Journal of Computer Science, Engineering and Information Technology, vol. 5, no. 1, 2015, pp. 07-26.
[8] Matarneh, Rami J. "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, 2009, pp. 1831-1837.
[9] Arora, Himanshi, et al. "An Improved CPU Scheduling Algorithm." International Journal of Applied Information Systems, vol. 6, no. 6, 2013, pp. 7-9.
[10] Babu.B, Sukumar, et al. "Efficient Round Robin CPU Scheduling Algorithm." International Journal of Engineering Research and Development, vol. 4, no. 9, Nov. 2012, pp. 36-42.
[11] Farooq, Muhammad U., et al. "An Efficient Dynamic Round Robin algorithm for CPU scheduling." 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), 2017.
[12] Dhumal, Ms. Rashmi A., et al. "Dynamic Quantum based Genetic Round Robin Algorithm." International Journal of Advanced Research in Computer and Communication Engineering, vol. 3, no. 3, Mar. 2014.
[13] Belwal, Monika, and Sanjay Kumar. "A Priority Based Round Robin CPU Scheduling Algorithm." International Journal of Computer Science and Information Technologies, vol. 8, no. 4, 2017.
[14] Asif, Md. R., et al. "Improved Performance of Round Robin CPU Scheduling Algorithm Using Non-preemptive SJF." International Journal of Scientific & Engineering Research, vol. 8, no. 12, 2017, pp. 1734-1738.
[15] Goel, Neetu, and Dr. R. "Performance Analysis of CPU Scheduling Algorithms with Novel OMDRRS Algorithm." International Journal of Advanced Computer Science and Applications, vol. 7, no. 1, 2016.
[16] Parashar, Mayank, and Amit Chugh. "Time Quantum based CPU Scheduling Algorithm." International Journal of Computer Applications, vol. 98, no. 3, 2014, pp. 45-48.
[17] Mohammad Dorgham, Omar H., and Dr. Mohammad O. Nassar. "Improved Round Robin Algorithm: Proposed Method to Apply SJF using Geometric Mean." INTERNATIONAL JOURNAL OF ADVANCED STUDIES IN COMPUTER SCIENCE AND ENGINEERING, vol. 05, no. 11, 2016.
[18] Shyam, Radhe, and Sunil K. 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, July 2014.
[19] Joshi, Rahul, and 2Shashi B. Tyagi. "Smart Optimized Round Robin (SORR) CPU Scheduling Algorithm." International Journal of Advanced Research in Computer Science and Software Engineering, vol. 5, no. 7, July 2015.

Md. Shafiul Alam Forhad completed B.Sc. Engg. degree in Computer Science and Engineering from Chittagong University of Engineering & Technology (CUET) in 2014 with outstanding result. He is now pursuing his M.Sc. Engg. degree in Computer Science and Engineering from the same university. His current research interests are Data Mining, Operating Systems, Cryptography, and Machine Learning. He was a lecturer in the Department of Computer Science and Information Technology of Patuakhali Science & Technology University. Now, he has been serving as a faculty member in the Department of Computer Science & Engineering (CSE), Chittagong University of Engineering & Technology (CUET), Bangladesh.

Md. Monowar Hossain completed B.Sc. Engg. degree in Computer Science and Engineering from Chittagong University of Engineering & Technology (CUET) in 2014 with outstanding result. He is now pursuing his M.Sc. Engg. degree in Computer Science and Engineering from Khulna University of Engineering & Technology (KUET). His current research interests are Natural Language Processing, Data Mining, Cryptography and Machine Learning. He was a solution delivery engineer at Systems Solutions & Development Technologies Ltd. Now, he has been serving as a faculty member in the Department of Computer Science & Engineering (CSE), Bangabandhu Sheikh Mujibur Rahman Science and Technology University (BSMRSTU), Gopalganj, Bangladesh.

Mrinmoy Das completed B.Sc. Engg. degree in Computer Science and Engineering from Chittagong University of Engineering & Technology (CUET) in 2014 with an excellent result. He is now obtaining his M.Sc. Engg. Degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology (BUET). His current research interests are Machine Learning, Security & Cryptography, Human Computer Interaction, and Data Mining. He is currently serving as a faculty member in the department of Computer Science and Engineering of Bangladesh University of Business and Technology (BUBT).