JAET

Journal homepage: http://jae-tech.com

Journal of Applied Engineering & Technology

ISSN : 2523-6032 ISSN-L : 2523-2924

Improving Round Robin Scheduling with Dynamic Time Quantum (IRRDQ)

Farheen Qazi[1*], Dur-e-Shawar Agha[1], Muhammad Naseem[1], Shahnila Badar[2], Fozia Hanif Khan[3]

[1] Department of Software Engineering, Sir Syed University of Engineering and Technology, Karachi, 7500, Pakistan.
[2] Department of Electrical Engineering, DHA Suffa University, Karachi, 75500, Pakistan
[3] Department of Mathematics, University of Karachi, Karachi, 75270, Pakistan

*Corresponding Author

DOI: https://doi.org/10.55447/jaet.07.02.115

Abstract: In the realm of real-time and multitasking environments, the Round Robin (RR) CPU scheduling algorithm is extensively utilized. Renowned for its fairness and avoidance of process starvation, RR allocates a specific time quantum to each process. However, the frequent context switching between processes during CPU scheduling, owing to the short time quantum, can adversely impact system performance by increasing average waiting time and response waiting time. To address these concerns, this paper introduces "Improving Round Robin Scheduling Using Dynamic Time Quantum (IRRDQ)," a modified version of RR aimed at diminishing turnaround time, average waiting time, and context switching. In our proposed approach, we arrange all incoming processes based on minimum burst time and dynamically assign an optimal time quantum to each process using the Shortest Job First (SJF) algorithm. The computation of time quantum is tailored to the burst times of individual processes. Through experimental demonstrations, we showcase the significant performance improvement of our algorithm compared to RR and other existing algorithms.

Graphical Abstract:

Keywords: Round Robin (RR), Context switching, Average waiting time, Average turnaround time, Shortest Job First (SJF).

1. Introduction

CPU scheduling is the most vital task performed by an operating system [1], [2]. Scheduling is also called the heart of a computer system; it deals with the allocation of resources of computers among

multiple processes. For the execution of any process, a set of resources are required that are allocated to it by the CPU. Numerous processes are arriving in the specified manner, and various scheduling techniques are implemented to enhance execution efficiency, reduce waiting time and turnaround time, while maximizing CPU utilization. A process can exist in one of five fundamental states: new, ready, running, waiting (blocked), and exit. A process migrates throughout its lifespan between different scheduling queues by different schedulers until it gets terminated. These scheduling queues are:

1.1 Ready Queue

All the ready processes to execute are placed in this queue and waiting for the response of the CPU.

1.2 Input/output Queue

This queue contains all the processes waiting for an I/O response. The processes must be selected for scheduling in a definite manner by operating the system from the queues mentioned above. Scheduling algorithms are classified into two broad categories [3]:

- Preemptive
- Non-Preemptive

2. CPU Performance Parameters

Various criteria can measure the performance of the CPU scheduling algorithms. The main goal of any scheduling algorithm is to meet the following criteria [4]:

2.1 CPU Utilization

It is the mediocre division of time. The range of CPU utilization is from 0 to 100. During the execution of processes, the CPU is not free; the processor is as busy as possible.

2.2 Throughput

The amount of work the CPU does in a unit of time (period). The higher the number of processes entertained by the system, the more work is done by the system.

2.3 Waiting Time

Refers to the process which waits in the ready queue, not the time required for process execution or I/O completion. The process is waiting to be assigned to the CPU.

2.4 Turnaround Time

It is also called the over-all time of a process. It is the intermission between the submission time of a process to the completion time of that process.

2.5 Response Time

This time of the scheduling should be low. It is the first response time of the request from the time of submission.

2.6 Fairness

Make sure that all the processes share the CPU equally; no process is in a state of starvation. Provide an equal opportunity for the execution of all processes.

3. Existing CPU Scheduling Algorithms

CPU scheduling has four popular algorithms. These algorithms choose which process is allocated to the CPU from the ready queue. Each algorithm has its pros and cons. The algorithms for CPU scheduling are as follows:

3.1 First Come, First Serve (FCFS)

The algorithm is based on the FIFO (First in First Out) queue rule. The allocation of processes to the CPU is according to the arrival of processes which are waiting in ready queue. Once the CPU is allocated to the process, it removes from the ready queue. It never gives up until and unless the process has completed all of its activities or execution because this algorithm is non-preemptive.

3.2 Shortest Job First (SJF)

The algorithm is based on the length of the CPU burst or execution time. CPU allocation to the processes concerning the shortest burst time. If two or more processes have same burst time, then the FCFS algorithm will break the tie between them [5]. The SJF algorithm is categorized into two different schemes:

3.2.1 SJF Preemptive

When a process currently running on the CPU and interrupted by a new arriving process with less CPU burst time, the currently running process returns to the ready queue, and the CPU entertains the interrupted process.

3.2.2 SJF Non-Preemptive

This algorithm does not allow interruption in the running. Once the execution of a running process is executed, the CPU will allocate to the next process.

3.3 Priority Scheduling (PS)

In the PS algorithm, processes are assigned CPU time based on their associated priority values. The lowest numerical value corresponds to the highest priority, and the process with the maximum priority is granted CPU allocation. In cases where two or more processes share the same priority, ties are resolved using the FCFS algorithm.

3.4 Round Robin (RR)

The basic motive for designing RR is equal time sharing for all arrived processes. A defined time quantum (small unit of time) is assigned to all processes. These processes are executed in FCFS fashion with equal time distribution. If the cycle of processes concludes and certain processes still have remaining burst time, the execution cycle is reiterated for all processes sharing the same time quantum value until each process completes its execution [6].

4. Literature Review

In recent years, research has been performed on operating system CPU scheduling algorithms. These works are the extension of already existing CPU scheduling algorithms. Authors have developed their models in their articles. We have discussed some articles related to CPU scheduling algorithms by different authors. The Round Robin Algorithm is rendered and the waiting time, turnaround time, and number of context switches are reduced by the algorithm proposed in [7]. First, the mean of the execution times of all the arrived processes is taken in order to calculate the time quantum. The procedures are then carried out. It recalculates the time quantum for those processes with remaining burst time if some processes execute their burst time completely. In [8], the author first arranges all of the processes in ascending manner and then uses the first cycle's execution time as a time quantum. Rearranged all the processes in ascending order, following the completion of the first cycle, which results in a new time quantum for the processes' remaining execution time. Because of this calculation, they secure improved results than the first Cooperative Calculation. An improved Round Robin algorithm known as ERRBTQ (Enhanced Round Robbin with Burst-Time Based Time Quantum) was proposed by the authors in this study [9]. After determining the median of each

process' burst time, time quantum is used to carry out the processes; It will execute the same process if any of the processes run and the remaining burst time is less than the time quantum. In the Modulo-Based Round Robin Algorithm [10], an ideal approach to ranking processes and locating the time quantum was proposed. The time quantum is the average of all the execution times for the given processes. Taking each process's modulo with the calculated time quantum and giving priority to the process with the smallest value; consequently, the turnaround time, average waiting time, and number of context switches are reduced by this algorithm. For choosing or calculating a time quantum, the modified Round Robin Algorithm [11] offers two distinct selection criteria. Calculate the average of all the burst times if the number of processes are even, but if the number of processes is odd select the quantum value as middle value. Each process with modified quantum time; It will carry out the same procedure if the process's remaining burst time is less than the quantum time. Priority Based Round Robin CPU Scheduling Using Dynamic Time Quantum [12] was the proposed method, which eliminates the issue of context switches and speeds up turnaround and waiting times. Take the average execution time of all the arrived processes to determine the dynamic time quantum. It executes two of the processes with the highest and second highest priority out of all of them for the calculated time quantum. Rehash these tasks till the finish of the execution season of the multitude of cycles. In this paper proposed a precautionary and non-preplanned nature calculation [13]. Condition factor (f) is a new factor that is calculated by adding the arrival time and execution time of the given processes in this algorithm. Each process is given this factor 'f,' which puts them in the ready queue and arranges them in decreasing order based on the factor 'f.' The process with the shortest value of the factor, "f," runs first, followed by the process with the next shortest value of the factor. Throughput and CPU utilization are both increased, and waiting, turnaround, and response times are reduced, by this algorithm. The round-robin algorithm was the focus in [14] there are three stages to this algorithm. The process's execution time is determined by placing all values in ascending order in the first stage. The last stage selects a process and assigns a quantum up to one smart time quantum from the ready queue. The second stage than calculates the mean burst time of all given processes. Continue the process that is currently running and compare burst time if value of burst time is less than one; once the process is finished, it returns to stage three. Context switching, waiting time, and turnaround time are also reduced by this algorithm. This paper proposed an efficient CPU scheduling preemptive nature algorithm that places the processes in ascending order in a ready queue. Every process is given a brief amount of time, or time quantum. The given time slice is followed by each process as it executes; If a process's time slice has run out, it will be moved to the end of queue, where the CPU will start the next process from the ready queue. The ready queue resembles a circular queue in this scenario. This algorithm outperforms the current CPU scheduling algorithm in terms of turnaround time and waiting time [15]. Another planning calculation, SJRR, was proposed by Rakesh Patel. A preemptive and round-robin mechanism served as the foundation for this algorithm. They arrange all of the processes according to minimum to maximum order of the burst time in a ready queue in an effort to reduce both the average waiting time and the turnaround time. Time quantum (TQ) is the process with the shortest burst time out of all of them [16]. Author introduced another cooperative planning calculation called superior cooperative effort [17]. They organized every one of the cycles in the rising request concerning the burst time and allocated an ideal time quantum to the cycles. They came up with a method that helped cut down on waiting times, turnaround times, and context switching.

5. Research Methodology

Our methodology is not to change the philosophy of the original Round Robin Algorithm; rather, we have added two more steps to this algorithm to achieve better results than the original one. Our proposed approach comprises Round Robin Algorithm (RR), Shortest Job First (SJF Non-Preemptive), and dynamically calculated time quantum for RR. The proposed algorithm is represented in Figure 1 and overall flow is represented in Figure 2.

In traditional Round robin processes are execute using First Come First Serve (FCFS) technique. In our proposed algorithm we execute processes in Shortest Job First, i-e. the process which has least execution time will execute first. Our proposed algorithm uses Dynamic Time Quantum which is also used in Web Server Scheduling in the context of web servers, where multiple user requests (processes) must be served concurrently, the IRRDQ principles have been applied in a practical context. The dynamic adjustment of the time quantum in scheduling algorithms becomes crucial to ensure efficient processing of user requests and optimal utilization of system resources. Let's consider a case study where the IRRDQ principles are applied in web server scheduling.

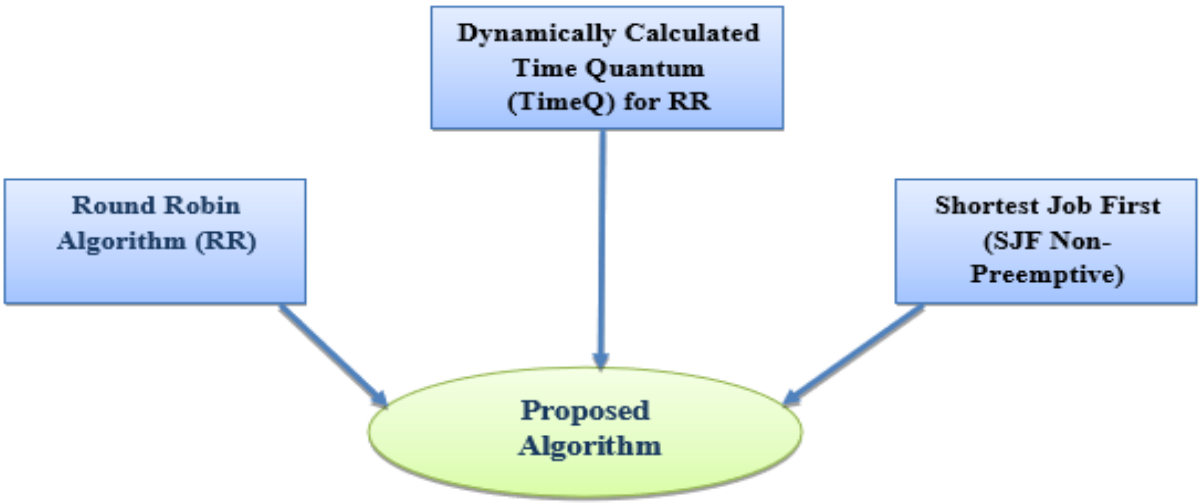

Fig. 1 - Round Robin Using Dynamic Time Quantum

For scheduling Round Robin, our proposed algorithm (IRRDQ) is effective. We compare our newly proposed algorithm to a few other algorithms that are already in use to determine its effectiveness and efficiency, such as Self-Adjustment Time Quantum in Round Robin Algorithms (SARR) [19] and SJRR CPU Scheduling Algorithm [16] are all variations of the round robin algorithm (RR). With some parameters, we compare these algorithms. The process arrangement, time quantum evaluation method or procedure, waiting time, and turnaround time are all compared in Table 1. We played out some experiments for trial results displayed in segments 6.1, 6.2, and 6.3.

5.1 Assumptions

In our proposed algorithm, we assume the following points:

- We consider an empty ready queue at the beginning.
- All arrived processes are presumed to have a uniform arrival time, set at Zero (0).
- All the arrived processes have their respective burst time.

5.2 Input

- All the processes which we want to execute.
- Every incoming process is associated with a specific burst time.

5.3 Output

- Dynamic Time Quantum (TimeQ) / Time Slice.
- Average Waiting Time.
- Average Turnaround Time.
- The number of Context Switches.

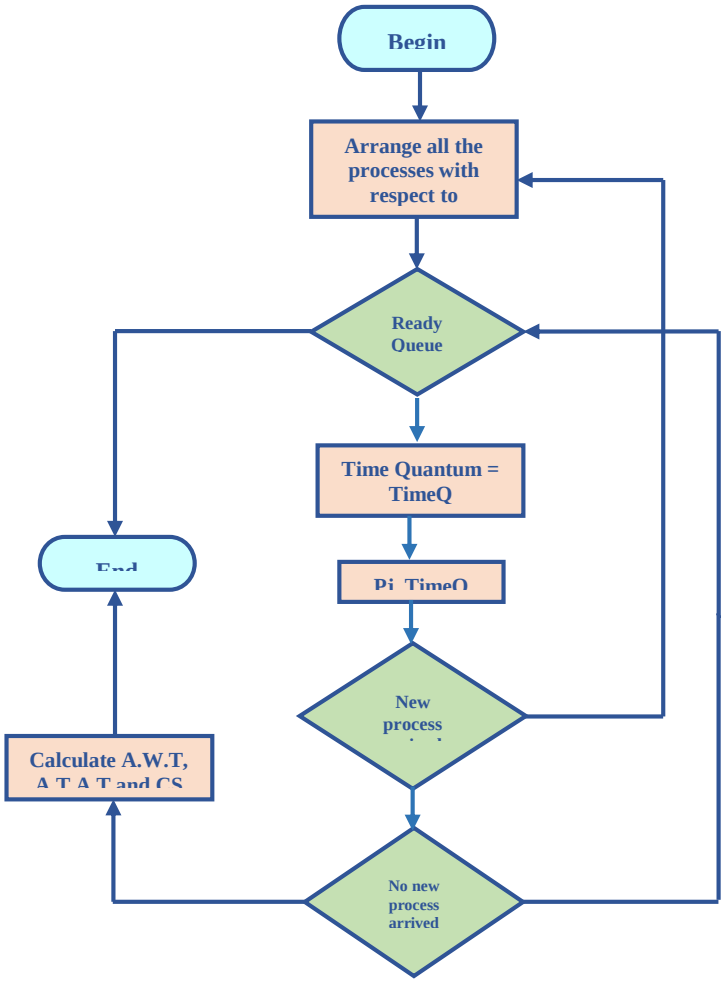

Fig. 2 - Flow Chart of Proposed Algorithm

Algorithm : IRRDQ (proposed algorithm)

1 **While**(Ready Queue != Null)

2 **Sort** all the processes with respect to the shortest job first (SJF)
3 **Calculate** Mean from step 2
4 **Calculate** the Combine Time (C.T = Highest Burst Time + Lowest Burst Time)
5 **Calculate** Time Quantum = TimeQ (TimeQ = square root (Mean+C.T))
6 **Assign** TimeQ to each process:
P[j] → TimeQ
6 **Take** next round for the remaining burst time (B.T) of the processes

7 **if** (new process arrived and B.T != Null)
8 **Move** toward step 5,
9 **else if** (no new process arrived and B.T != 0)
10 **End if**
11 **end**

12 **End While**

5.4 Parameters of Proposed Algorithms

5.4.1 Combine Time (C.T)

The Combine Time (C.T) is calculated by finding the sum of the highest burst time and the lowest burst time among a set of processes in a multitasking or multiprocessing environment. In a multitasking system, processes are typically assigned burst times, which represent the amount of time it takes for a process to execute without being interrupted. The highest burst time refers to the process with the longest execution time, while the lowest burst time refers to the process with the shortest execution time. The Combine Time is used in various scheduling algorithms to make decisions about the order in which processes are executed. By adding the highest burst time and the lowest burst time, the system can get a sense of the range of execution times among the processes, which can help in making efficient scheduling decisions.

5.4.2 Time Quantum (TimeQ)

To calculate the Time Quantum (TimeQ) using the formula TimeQ = √(Mean + C.T), the following steps can be followed:
The mean (average burst time) of the processes is calculated by summing up all the burst times and then dividing by the number of processes.
As mentioned earlier, the Combine Time (C.T) is computed by adding the highest burst time and the lowest burst time among a set of processes. Once the Mean and C.T values are obtained, substitute them into the formula TimeQ = √(Mean + C.T).

Table 1 - Parameters Comparison for Algorithms

Parameters / ***Algorithms***	***Arrangement of Processes***	***Time Quantum***	***Waiting Time***	***Turnaround Time***
IRRDQ	Arrange all the processes into first come first serve (FCFS) order	***TimeQ*** = $\sqrt{Mean + C.T}$ *Mean= $\sum BurstTime$ (all arrived processes) *C.T=Highest Burst Time + Lowest Burst Time	***Waiting Time = Start Time – Arrival Time*** **IRRDQ < RR, IRR, SARR, SJRR**	***Turnaround Time = Waiting Time + Burst Time*** **IRRDQ < RR, IRR, SARR, SJRR**
RR	Arrange all the processes into first come first serve (FCFS) order	***TQ = Random Value***	***Waiting Time = Start Time – Arrival Time*** **RR > IRRDQ, IRR, SARR, SJRR**	***Turnaround Time = Waiting Time + Burst Time*** **RR > IRRDQ, IRR, SARR, SJRR**
IRR	Arrange all the processes into first come first serve (FCFS) order	***TQ = Random Value*** (If a process is executed following the time quantum (TQ), and the remaining burst time of the same process is less than the TQ value, the CPU will execute the same process again)	***Waiting Time = Start Time – Arrival Time*** **IRR < RR, SARR, SJRR**	***Turnaround Time = Waiting Time + Burst Time*** **IRR < RR, SARR, SJRR**
SARR	Organize the provided processes in ascending order based on their Burst Time.	***TQ = Median*** (Determine the median value from the burst times of the processes that have arrived)	***Waiting Time = Start Time – Arrival Time*** **SARR < RR, SJRR**	***Turnaround Time = Waiting Time + Burst Time*** **SARR < RR, SJRR**

SJRR	Sort the given processes in ascending order based on their Burst Time.	***TQ = Value of smallest burst time process***	***Waiting Time = Start Time – Arrival Time*** **SJRR < RR**	***Turnaround Time = Waiting Time + Burst Time*** **SJRR < RR**
ARRS	Arrange all the processes into first come first serve (FCFS) order	***TQ = Random Value*** **(abs(BT[i] - TQ) <= (TSH× TQ))** (This algorithm employs a specified threshold value and conducts a comparison based on a given formula. If the condition evaluates as true, the quantum is set equal to the burst time of the desired process; otherwise, it adopts a randomly generated quantum value)	***Waiting Time = Start Time – Arrival Time*** **ARRS < RR, SJRR**	***Turnaround Time = Waiting Time + Burst Time*** **ARRS < RR, SJRR**

6. Results and Discussion

We have analyzed our proposed IRRDQ algorithm against RR, IRR [18], SARR [19], SJRR [16], and ARRS [20]. The outcomes of our algorithm were compared with those of the existing Round Robin (RR), Improved Round Robin (IRR), Self-adjustment Round Robin, and SJRR in each of the three scenarios. For RR, IRR, and ARRS algorithms, a fixed (static) time quantum (TQ) of 40 was utilized. SJRR determines the quantum time by selecting the shortest burst time, while SARR uses the median of processes' burst times as the time quantum (TQ).

Case 1: CPU Burst Time According to Shortest Burst Time - Let's consider the system currently having five processes (P1, P2, P3, P4, and P5) with arrival time = 0 (zero) and burst time (27, 54, 79, 93, and 140), as shown in Tables 2 and 3. The output of case 1 algorithms is shown in Fig. 3 and 4 in the form of Gantt charts.

Table 2 - Input Data (Case 1)

Process #	Arrival Time	Burst Time
P1	0	27
P2	0	54
P3	0	79
P4	0	93
P5	0	140

Table 3 - Algorithms Comparison (Case 1)

Algorithms	*Time Quantum (TQ)*	*Average Waiting Time (A.W.T)*	*Average Turnaround Time (A.T.A.T)*	*Context Switching (C.S)*
IRRDQ	114	104.2	182.8	5
RR	40	218.2	296.8	11
IRR	40	112.2	190.8	7
SARR	79	120	198.6	7
SJRR	27	158.2	231.4	13
ARRS	40	136.2	214.8	9

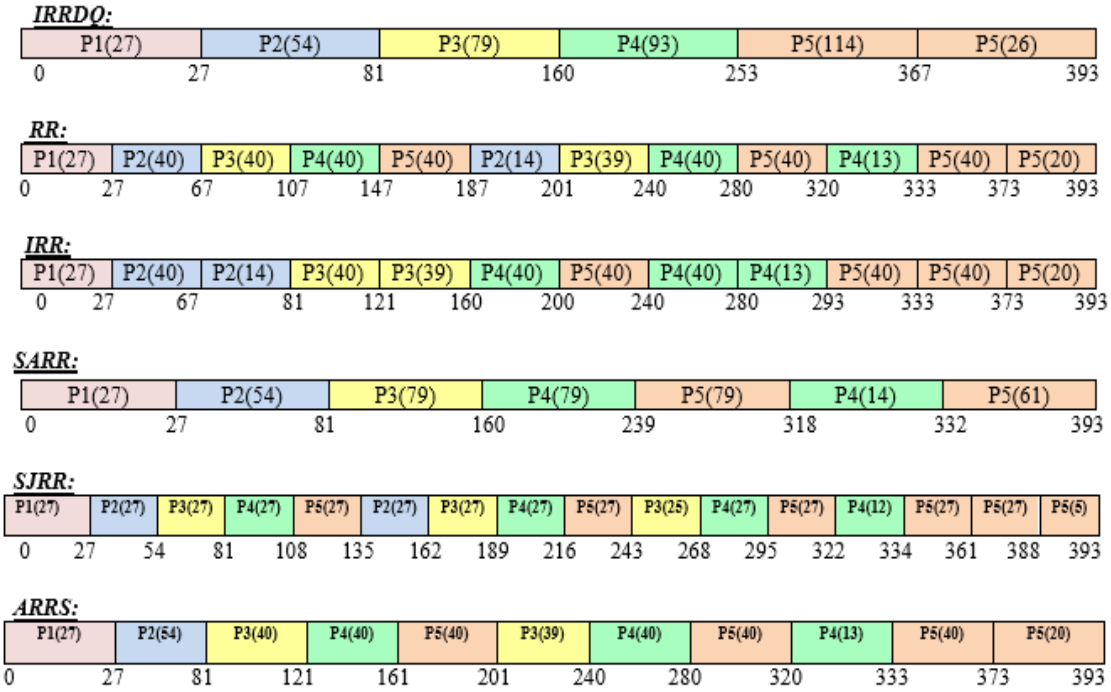

Fig. 3 - Gantt Charts of Case-01 Algorithms

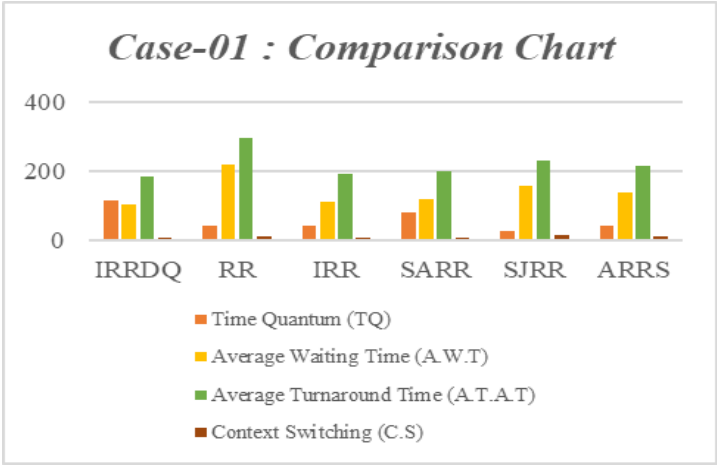

Fig. 4 - Comparison between Quantum Time, Avg. Waiting Time and Avg. Turnaround Time and Context Switching of Case 1 Algorithms

Case 2: CPU Burst Time According to Highest Burst Time - Let's consider the system currently having five processes (P1, P2, P3, P4, and P5) with arrival time = 0 (zero) and burst time (140, 93, 79. 54, and 27), as shown in Tables 4 and 5. The output of case 2 algorithms is shown in Fig. 5 and 6 in the form of Gantt charts.

Table 4 - Input Data (Case 2)

Process #	Arrival Time	Burst Time
P1	0	140
P2	0	93
P3	0	79
P4	0	54
P5	0	27

Table 5 - Algorithms Comparison (Case 2)

Algorithms	Time Quantum (TQ)	Average Waiting Time (A.W.T)	Average Turnaround Time (A.T.A.T)	Context Switching (C.S)
IRRDQ	114	104.2	182.8	5
RR	40	237.2	315.8	12
IRR	40	189	267	8
SARR	79	245	323.6	7
SJRR	27	163.6	242.2	13
ARRS	40	213.6	292.2	10

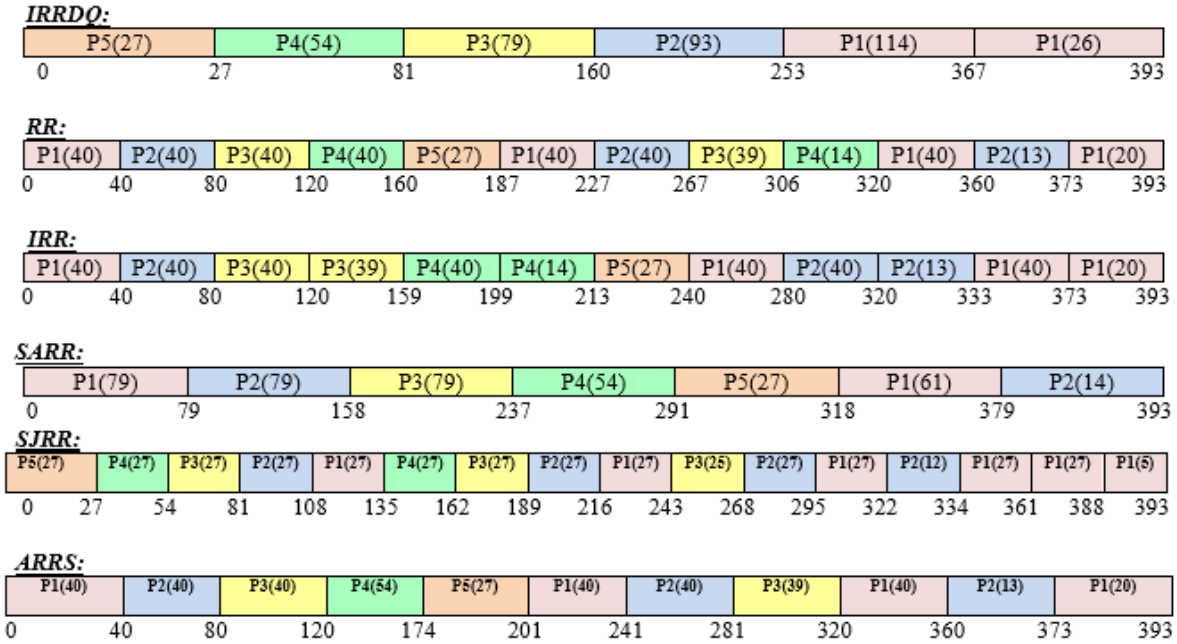

Fig. 5- Gantt Charts of Case-02 Algorithm

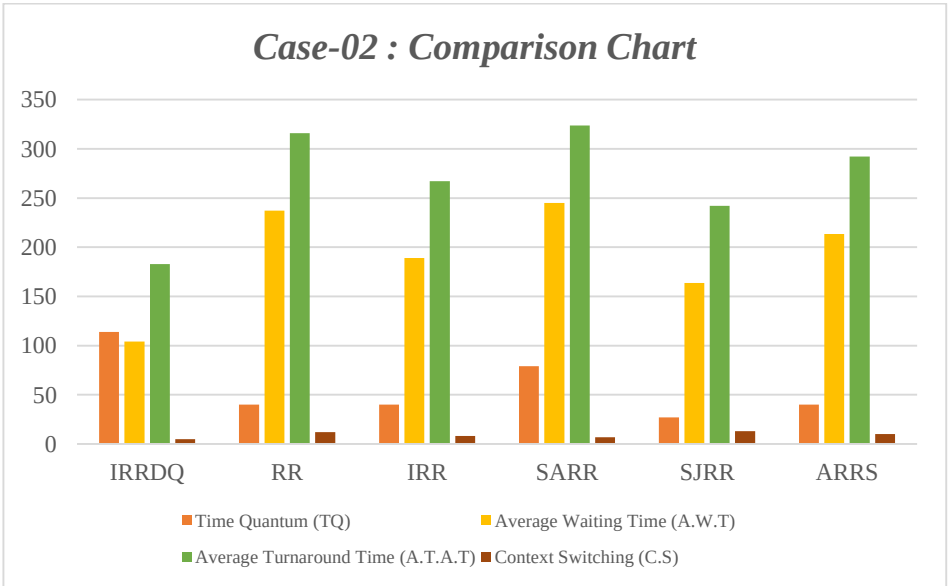

Fig. 6 - Comparison between Quantum Time, Avg. Waiting Time and Avg. Turnaround Time and Context Switching of Case 2 Algorithms

Case 3: CPU Burst Time in Random Order - Let's consider the system currently having five processes (P1, P2, P3, P4, and P5) with arrival time = 0 (zero) and burst time (93, 140, 79. 27, and 54) as shown in Tables 6 and 7. The output of case 3 algorithms is shown in Figure 7 and 8 in the form of Gantt charts.

Table 6 - Input Data (Case 3)

Process #	Arrival Time	Burst Time
P1	0	93
P2	0	140
P3	0	79
P4	0	27
P5	0	54

Table 7 - Algorithms Comparison (Case 3)

Algorithms	*Time Quantum (TQ)*	*Average Waiting Time (A.W.T)*	*Average Turnaround Time (A.T.A.T)*	*Context Switching (C.S)*
IRRDQ	114	104.2	182.8	5
RR	40	221.2	299.8	11
IRR	40	175.6	254.2	8
SARR	79	230	308.6	7
SJRR	27	158.2	236.8	13
ARRS	40	200.2	278.8	9

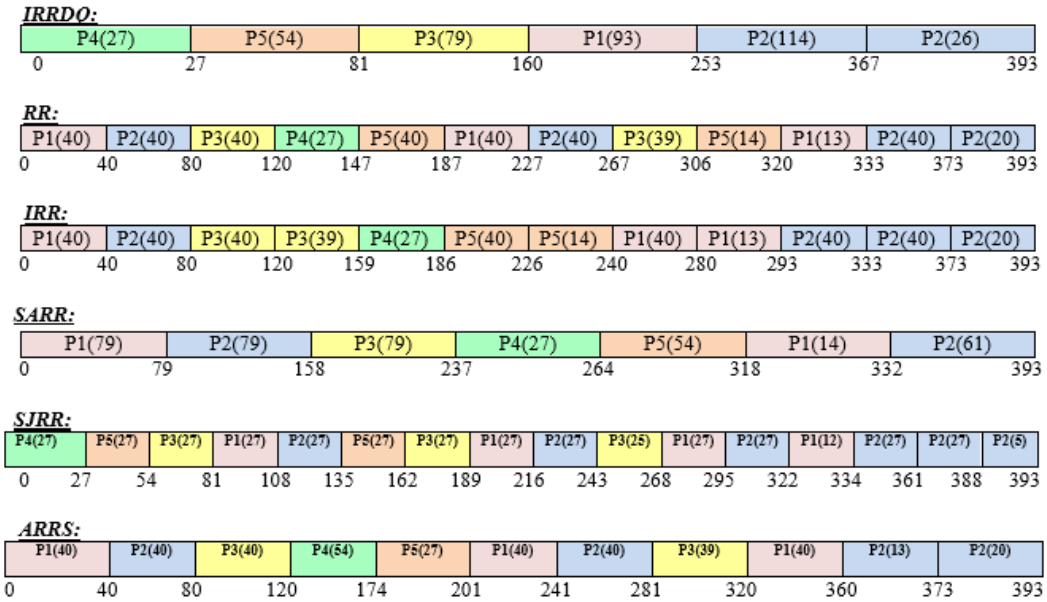

Fig. 7 - Gantt Charts of Case 3 Algorithms

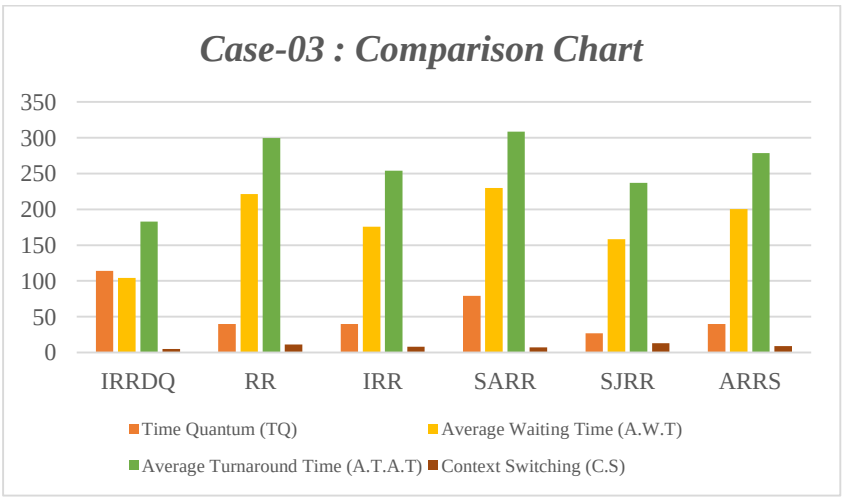

Fig. 8 - Comparison between Quantum Time, Avg. Waiting Time and Avg. Turnaround Time and Context Switching of Case 3 Algorithms

7. Conclusion

The presented exploratory results highlight the advanced performance of the IRRDQ algorithm in comparison to conventional Cooperative effort (R.R.) Calculation, Enhanced Cooperative effort (IRR) Calculation, SJRR microprocessor Planning Calculation, Self-Adjusting Time Quantum in Round Robin (SARR), and Adjustable Round Robin Scheduling Algorithm (ARRS) Calculation. Three distinct cases discussed in the experimental findings all demonstrate that IRRDQ meets optimization criteria by achieving the shortest possible waiting time (Avg. W.T.), minimizing turnaround time (Avg. T.A.T.), and minimizing context switches (C.S.). Using the proposed algorithm as a foundation, potential future improvements can be explored.

8. Future Enhancement

Research and improvements on the proposed IRRDQ process scheduling algorithm is possible in many ways, if we investigate methods to dynamically adjust the priorities of processes in real-time based on their resource needs and system conditions. This could include incorporating feedback mechanisms that adaptively modify process priorities to ensure fair and efficient resource utilization. Also consider optimizing the interprocess communication mechanism in the IRRDQ algorithm. This could involve developing more efficient protocols or exploring advanced communication techniques, such as shared memory or event-driven architectures, to minimize overhead and improve overall system performance. Integration with energy-aware scheduling: Explore the integration of energy-aware scheduling techniques into the IRRDQ algorithm. This could involve considering energy consumption as an additional parameter in the scheduling decisions to promote energy-efficient resource utilization and prolong the battery life of mobile devices. With further research and experimentation, there is great potential for refining and expanding the IRRDQ process scheduling algorithm to meet the evolving needs of complex computing systems.

References

[1]. Silber Schatz, Galvin and Gagne, *Operating systems concepts*, 9th ed. Wiley, 2018.

[2]. Mutlag, A. A., Abd Ghani, M. K., Mohammed, M. A., Lakhan, A., Mohd, O., Abdulkareem, K. H., & Garcia-Zapirain, B. (2021). Multi-agent systems in fog–cloud computing for critical healthcare task management model (CHTM) used for ECG monitoring. *Sensors*, *21*(20), 6923.

[3]. Harki, N., Ahmed, A., & Haji, L. (2020). CPU scheduling techniques: A review on novel approaches strategy and performance assessment. *Journal of Applied Science and Technology Trends*, *1*(2), 48-55.

[4]. Zouaoui, S., Boussaid, L., & Mtibaa, A. (2019). Priority based round robin (PBRR) CPU scheduling algorithm. *International Journal of Electrical & Computer Engineering (2088-8708)*, *9*(1).

[5]. Harki, N., Ahmed, A., & Haji, L. (2020). CPU scheduling techniques: A review on novel approaches strategy and performance assessment. *Journal of Applied Science and Technology Trends*, *1*(2), 48-55.

[6]. Anselmi, J. (2019). Combining size-based load balancing with round-robin for scalable low latency. *IEEE Transactions on Parallel and Distributed Systems*, *31*(4), 886-896.

[7]. Tajwar, M.M., Pathan, M.N., Hussaini, L. and Abubakar, A., "CPU scheduling with a round robin algorithm based on an effective time slice". *Journal of Information processing systems*, vol. *13, no.* 4, pp.941-950, 2017.

[8]. Nizam, F., Ahmed, S., Kumar, P., Tsetse, A., Kumar, R., Shanker, B., & Sohu, N. (2023). Enhancing Physical Layer Security in MIMO Systems through Beamforming and Artificial Noise Techniques. *Journal of Independent Studies and Research Computing*, *21*(2), 20-24.

[9]. Berhanu, Y., Alemu, A. and Mishra, M.K., "Dynamic time quantum based round robin CPU scheduling algorithm". *International Journal of Computer Applications*, vol. 167, no. 13, pp.48-55, 2017.

[10]. Mostafa, S. M., Idris, S. A., & Kaur, M. (2022). ATS: A Novel Time-Sharing CPU Scheduling Algorithm Based on Features Similarities. *Computers, Materials & Continua*, *70*(3).

[11]. Joshi, A. and Gosswami, S., Modified round robin algorithm by using priority scheduling. *Advances in Computational Sciences and technology*, vol. 10, no. 6, pp.1543-1549, 2017.

[12]. *Nabi, AA, Bablani, A., Ali, M., Tunio, FH, Mukhi, A., & Soho, NU (2023). The Adoption of RFID Technology and Its Influence on Customer Satisfaction in Pakistan's Retail Industry: A Case Study of LuckyOne Mall. Research Letters, 1*(1), 27-34.

[13]. Mittal, N., Garg, K. and Ameria, A., "A paper on modified round robin algorithm". *International Journal of Latest Technology in Engineering, Management & Applied Science*, vol. 4, no. 11, pp.93-98, 2015.

[14]. Siddiqui, M. M. A., Sohu, M. N. U., & Zardari, M. H. A. (2023). Cyber Security and quality education: Recent Cyber-Attacks as a Challenge to National Economic Security. *International Research Journal of Management and Social Sciences*, *4*(1), 32-52.

[15]. Lulla, D., Tayade, J. and Mankar, V., "Priority based round robin cpu scheduling using dynamic time quantum". *IJETT*, vol. 2, no. 2, 2015.

[16]. Manwani, A., Dhirani, J., Baloch, A., & Wasan, R. H. (2022). Design of Voice-Controlled Robot Vehicle. *Journal of Applied Engineering & Technology (JAET)*, *6*(1), 40-47.

[17]. Bandarupalli, S.B., Nutulapati, N.P. and Varma, P.S., "A Novel CPU Scheduling Algorithm–Preemptive & Non-Preemptive". *International Journal of Modern Engineering Research (IJMER)*, vol. 2, no. 6, pp.4484-4490, 2012.

[18]. Joshi, R. and Tyagi, S.B., "Smart optimized round robin (SORR) CPU scheduling algorithm". *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5, no. 7, pp.568-574, 2015.

[19]. Dewangan, B. K., Jain, A., & Choudhury, T. (2020). AP: Hybrid Task Scheduling Algorithm for Cloud. *Rev. d'Intelligence Artif.*, *34*(4), 479-485.

[20]. Patel, R. and Patel, M., "SJRR CPU scheduling algorithm". *Int J Eng Comput Sci*, *2*, pp.3396-3399, 2013.

[21]. Nayak, D., Malla, S.K. and Debadarshini, D., "Improved round robin scheduling using dynamic time quantum". *International Journal of Computer Applications*, vol. 38, no. 5, pp.34-38.

[22]. Mishra, M.K., "An improved round robin CPU scheduling algorithm". *Journal of Global Research in computer science*, vol. *3*, no. 6, pp.64-69, 2012.

[23]. Matarneh, R.J., 2009. Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes. *American Journal of Applied Sciences*, vol. 6, no. 10, pp.1831-1837.

[24]. Mostafa, S., & Amano, H. (2019). An adjustable round robin scheduling algorithm in interactive systems. *Information Engineering Express*, *5*(1), 11-18.