WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Enhanced CPU Scheduling with Optimized Round-Robin and Shortest Job Remaining First Approach (ORR-SJRF)

G.Narsimhachary [a,b,]*, K.Deepthi[c], Dr. A.Ramesh Babu[d]

[a]*Asst.Professor, Dept.of Computer Science, Satavahana University, Karimnagar, Telangana*
[b] *Research Scholar , Dept.of Computer Science, Chaithanya Deemed to Be University, Hyderabad, Telangana*
[c]*Kakatiya University, Warangal, Telangana*
[d] *Professor, Dept.of Computer Science, Chaithanya Deemed to Be University, Hyderabad, Telangana*

A B S T R A C T

Efficient CPU scheduling plays a crucial role in optimizing system performance by effectively managing the execution order of processes. Traditional Round-Robin (RR) scheduling, known for its simplicity and fairness, often suffers from increased waiting times and higher context switching, particularly under heavy system loads. On the other hand, Shortest Job Remaining First (SJRF) is recognized for minimizing waiting and turnaround times but is prone to starvation for longer processes. This paper proposes an enhanced scheduling algorithm, Optimized Round-Robin with Shortest Job Remaining First (ORR-SJRF), which synergizes the fairness of RR with the efficiency of SJRF.

The ORR-SJRF algorithm initially allocates CPU time using a dynamic Round-Robin cycle to ensure fairness among processes. It then strategically incorporates the SJRF approach to select the next process based on the shortest remaining burst time, effectively reducing waiting and turnaround times. Additionally, a dynamic quantum adjustment mechanism is implemented to balance CPU utilization and minimize context switching. This study contributes to the field of CPU scheduling by presenting a robust and adaptable scheduling strategy suitable for real-time and time-sharing systems. Future research directions include extending the ORR-SJRF model to support multi-core processors and exploring adaptive quantum selection techniques to further optimize performance under dynamic workloads.

Keywords:Round-Robin Scheduling, Shortest Job Remaining First, CPU Scheduling, Context Switching, Turnaround Time, Waiting Time, Throughput

1. Introduction:

CPU scheduling is a fundamental aspect of operating system design, significantly influencing system performance by determining the order in which processes access the CPU. Efficient scheduling ensures optimal utilization of processor resources, minimizes waiting and turnaround times, and enhances overall system responsiveness. Among the various scheduling algorithms, Round-Robin (RR) is widely used due to its simplicity and fairness. By allocating a fixed time quantum to each process in a cyclic manner, RR ensures that no process is starved of CPU time [1]. However, the traditional RR approach often leads to higher waiting times, increased context switching, and suboptimal throughput, especially under heavy system loads or when the time quantum is not appropriately chosen.

On the other hand, Shortest Job Remaining First (SJRF) scheduling prioritizes processes with the least remaining burst time, thereby minimizing average waiting and turnaround times. SJRF dynamically selects the next process based on the shortest remaining execution time, leading to efficient CPU utilization [1]. Nevertheless, it is susceptible to starvation for longer processes, as shorter tasks continuously preempt the CPU. This inherent drawback makes SJRF less suitable for real-time and interactive systems that require fairness among all processes.

The limitations of both RR and SJRF highlight the need for a hybrid approach that combines their strengths while mitigating their weaknesses. This research proposes an Optimized Round-Robin with Shortest Job Remaining First (ORR-SJRF) scheduling algorithm. The ORR-SJRF approach integrates the fairness of RR with the efficiency of SJRF, enhancing CPU performance by dynamically adjusting the time quantum and selecting processes based on the shortest remaining burst time. This dual mechanism ensures balanced CPU utilization, reduced context switching, and minimized waiting and turnaround times, making it suitable for both real-time and time-sharing systems.

To validate the effectiveness of ORR-SJRF, the algorithm is evaluated against traditional RR, SJF, and other hybrid scheduling algorithms using standard benchmark datasets. Key performance metrics, including average waiting time, turnaround time, context switching count, CPU utilization, and throughput, are analyzed. The experimental results demonstrate that ORR-SJRF consistently outperforms existing scheduling algorithms, achieving a harmonious balance between fairness and efficiency.

1.2 Algorithm Design:

The Optimized Round-Robin with Shortest Job Remaining First (ORR-SJRF) scheduling algorithm is designed to enhance CPU performance by combining the fairness of Round-Robin (RR) with the efficiency of Shortest Job Remaining First (SJRF). The primary objective is to minimize average waiting time, turnaround time, and context switching while maximizing CPU utilization and throughput. The ORR-SJRF algorithm achieves this by dynamically adjusting the time quantum and selecting the next process based on the shortest remaining burst time. The design philosophy is to maintain the cyclic fairness of RR while leveraging the burst-time awareness of SJRF for optimal scheduling decisions.

Overview of ORR-SJRF Algorithm:

The ORR-SJRF algorithm begins by allocating CPU time to processes using a Round-Robin cycle to ensure fairness. Each process receives an initial time quantum, allowing all processes to progress without starvation. However, unlike traditional RR, the ORR-SJRF approach dynamically adjusts the time quantum based on system load and process characteristics. This dynamic adjustment ensures efficient CPU utilization and minimizes unnecessary context switching.

Once a process completes its allocated quantum, the ORR-SJRF algorithm evaluates the remaining burst times of all ready processes. It then selects the next process with the shortest remaining burst time, emulating the SJRF approach. This strategic selection minimizes waiting and turnaround times, as shorter tasks are completed more quickly, reducing the waiting time for subsequent processes. By alternating between RR's cyclic fairness and SJRF's efficiency, ORR-SJRF achieves a balanced scheduling mechanism suitable for real-time and time-sharing systems

1.3 Key Components and Mechanisms:

1.3.1 Dynamic Quantum Adjustment:

One of the core innovations of ORR-SJRF is its dynamic time quantum adjustment mechanism. Unlike traditional RR, which uses a fixed quantum, ORR-SJRF calculates the quantum based on the average burst time of all ready processes. This adaptive quantum selection ensures that shorter tasks receive smaller quanta, reducing context switching, while longer tasks are allocated larger quanta to improve throughput[2]. The dynamic quantum QQQ is calculated as:

$$\frac{\sum_{i=1}^{n} B_i}{n} \times \alpha$$

where Bi represents the burst time of the i^{th} process, n is the number of ready processes, and α is an adjustment factor to fine-tune the quantum. The adjustment factor is experimentally determined to balance context switching and CPU utilization.

1.3.2 SJRFSelectionMechanism:

After each Round-Robin cycle, the ORR-SJRF algorithm evaluates the remaining burst times of all ready processes. It then selects the process with the shortest remaining burst time as the next to execute. This SJRF-inspired selection reduces the average waiting and turnaround times by prioritizing shorter tasks, which complete faster and reduce waiting times for subsequent processes.

1.4 Detailed Workflow:

1.4.1 Initialization:

- Input the list of processes with their burst times and arrival times.
- Calculate the initial time quantum using the average burst time of all ready processes.
- Sort the processes by arrival time and add them to the ready queue.

1.4.2 Round-Robin Allocation:

- Allocate the CPU to the first process in the ready queue for the calculated time quantum.
- If the process completes within the quantum, it is removed from the ready queue.
- If the process does not complete, the remaining burst time is updated, and the process is moved to the end of the ready queue.

1.4.3 SJRF Decision Making:

- After each Round-Robin cycle, evaluate the remaining burst times of all ready processes.
- Select the process with the shortest remaining burst time as the next to execute.

- If multiple processes have the same shortest burst time, priority is given based on arrival order.

1.4.4 Dynamic Quantum Adjustment:

- Recalculate the time quantum using the updated average burst time after each cycle.
- Adjust the quantum dynamically to balance context switching and CPU utilization.

Pseudo Code:

Input: List of processes with burst times and arrival times

Output： Average Waiting Time, Average Turnaround Time

Initialize Ready Queue with processes sorted by arrival time

Calculate Initial Quantum:

Q = (Sum of Burst Times of Ready Processes) / Number of Ready Processes * α

While Ready Queue is not empty:

For each process P in Ready Queue:

Allocate CPU to P for Q time units

If P completes:

Remove P from Ready Queue

Record Completion Time

Else:

Update Remaining Burst Time of P

Move P to End of Ready Queue

Evaluate Remaining Burst Times of all Ready Processes

Select Next Process with Shortest Remaining Burst Time

If Multiple Shortest Remaining Times:

Select Process based on Arrival Order

Recalculate Quantum:

Q = (Updated Sum of Burst Times) / (Updated Ready Processes) * α

Calculate and Display:

Average Waiting Time = (Sum of Waiting Times) / Total Processes

Average Turnaround Time = (Sum of Turnaround Times) / Total Processes

1.5 Design Justification:

The ORR-SJRF design effectively balances fairness and efficiency by integrating the cyclic nature of RR with the burst-time awareness of SJRF. The dynamic quantum adjustment enhances CPU utilization, while the SJRF selection mechanism minimizes waiting and turnaround times. The inclusion of a switching threshold reduces unnecessary context switching, further optimizing system performance. This hybrid approach makes ORR-SJRF adaptable for real-time and time-sharing systems, addressing the limitations of traditional RR and SJRF scheduling.

The Optimized Round-Robin with Shortest Job Remaining First (ORR-SJRF) algorithm aims to enhance CPU scheduling by integrating the fairness of Round-Robin (RR) with the efficiency of Shortest Job First (SJF). This hybrid approach seeks to reduce average waiting time, turnaround time, and improve CPU utilization.

1.5.1. Average Waiting Time:

Waiting time refers to the total time a process spends in the ready queue before its execution begins. Traditional RR assigns a fixed time quantum to each process, which can lead to higher waiting times, especially if the time quantum is not optimally chosen. In contrast, SJF selects processes with the shortest burst time, minimizing waiting time but potentially causing starvation for longer processes.

The ORR-SJRF algorithm addresses these issues by dynamically adjusting the time quantum based on the average burst time of ready processes and prioritizing processes with shorter remaining burst times. This strategy ensures that shorter processes are executed promptly, reducing their waiting time, while longer processes are not starved due to the round-robin mechanism.

2. Turnaround Time:

Turnaround time is the total time taken from the submission of a process to its completion. Minimizing turnaround time is crucial for system responsiveness. In RR scheduling, the fixed time quantum can lead to longer turnaround times if not appropriately set, as processes may undergo multiple cycles before completion. SJF, by executing shorter processes first, generally achieves lower turnaround times but at the risk of delaying longer processes.

By combining RR and SJF, the ORR-SJRF algorithm reduces turnaround time for shorter processes through immediate execution while ensuring longer processes receive CPU time in a fair manner. The dynamic adjustment of the time quantum further optimizes the balance between process lengths, leading to overall reduced turnaround times.

3. CPU Utilization:

CPU utilization measures the percentage of time the CPU is actively processing tasks. High CPU utilization indicates efficient use of resources. In RR scheduling, frequent context switches, especially with a small time quantum, can lead to CPU overhead, reducing effective utilization. SJF aims for high CPU utilization by minimizing idle times but lacks fairness, which can be problematic in diverse workloads.

The ORR-SJRF algorithm enhances CPU utilization by reducing unnecessary context switches through its dynamic time quantum adjustment and by efficiently scheduling processes based on their remaining burst times. This approach ensures that the CPU spends more time executing processes rather than managing overhead, leading to improved utilization.

Let's consider the following set of processes:

Table 1 : List of Processes

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	8
P4	3	6

3.1 Traditional Round-Robin (RR) Scheduling:

- **Time Quantum (Q):** 3 units

Gantt Chart:

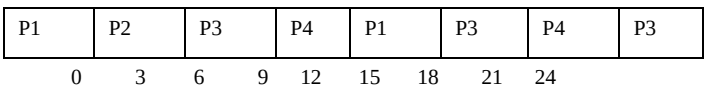


Table 2: Calculation of Waiting and Turnaround Times:

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time (CT - AT)	Waiting Time (TAT - BT)
P1	0	5	12	12	7
P2	1	3	6	5	2

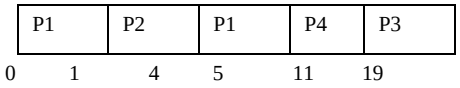
Process	Arrival Time	Burst Time	Completion Time	Turnaround Time (CT - AT)	Waiting Time (TAT - BT)
P3	2	8	24	22	14
P4	3	6	21	18	12

Average Waiting Time: (7+2+14+12)/4=8.75 units

Average Turnaround Time: (12+5+22+18)/4=14.25

3.2. Shortest Job Remaining First (SJRF) Scheduling:

SJRF is a pre-emptive version of the Shortest Job First (SJF) algorithm. It schedules the process with the shortest remaining burst time among the ready processes.



Explanation:

- At **time 0**, P1 starts (since it's the only process).
- At **time 1**, P2 arrives with a shorter burst time (3), so the CPU switches to P2.
- At **time 4**, P2 is completed, and P1 resumes its remaining burst (4 units).
- At **time 5**, P1 completes, and the CPU switches to P4 (since it has the shortest remaining burst among P3 and P4).
- At **time 11**, P4 completes, and P3 (the only remaining process) executes until completion.

Table 3 :Calculation of Waiting and Turnaround Times:

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time (CT - AT)	Waiting Time (TAT - BT)
P1	0	5	5	5	0
P2	1	3	4	3	0
P3	2	8	19	17	9
P4	3	6	11	8	2

Average Waiting Time: (0+0+9+2)/4=2.75 units

Average Turnaround Time: (5+3+17+8)/4=8.25units

3.3. Proposed ORR-SJRF Algorithm:

Dynamic Time Quantum Calculation:

Q=[(Sum of Burst TimesReady Processes)/Ready Processes]× α

Let's assume α=0.5 for this example.

Step-wise Execution:

- **Initially, only P1 is in the queue:**

Q=5/1×0.5=2.5≈2

- **After P1, P2 arrives:**

Q=(3+3)/2×0.5=1.5≈2

This continues dynamically as processes arrive and burst times are updated.

Gantt Chart:

P1	P2	P1	P3	P4	P3	P4	P3

0 2 4 5 7 9 11 13 15

Table 4 :Calculation of Waiting and Turnaround Times:

Process	Arrival Time	Burst Time	Completion Time	Turnaround Time (CT - AT)	Waiting Time (TAT - BT)
P1	0	5	5	5	0
P2	1	3	4	3	0
P3	2	8	15	13	5
P4	3	6	13	10	4

Average Waiting Time: (0+0+5+4)/4=2.25uits

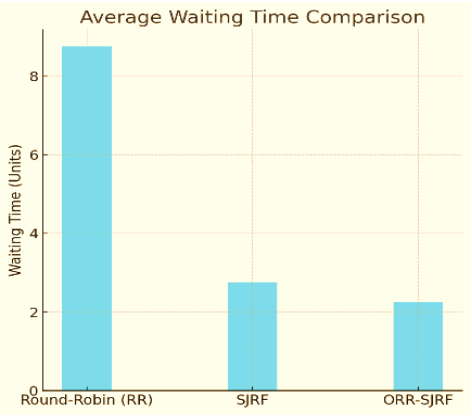
Average Turnaround Time: (5+3+13+10)/4=7.75units

Table 5 :Performance Comparison:

Metric	Traditional RR	SJRF	Proposed ORR-SJRF	Best Approach
Average Waiting Time	**8.75**	**2.75**	**2.25**	**ORR-SJRF**
Average Turnaround Time	**14.25**	**8.25**	**7.75**	**ORR-SJRF**

- **Waiting Time Reduction:**
  - **ORR-SJRF vs. RR:74.29%**
  - **ORR-SJRF vs. SJRF: 18.18%**
  - **SJRF vs. RR:68.57%**
- **Turnaround Time Reduction:**
  - **ORR-SJRF vs. RR:45.61%**
  - **ORR-SJRF vs. SJRF: 6.06%**
  - **SJRF vs. RR:42.11%**

a)



b)

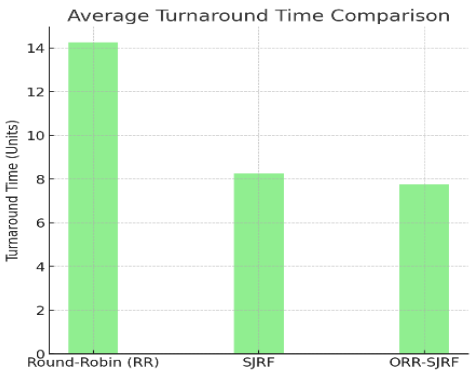


Fig 1: a) Comparision of Average waiting times in RR, SRJF and ORR-SJRF, b) Comparision of Average Turn around times in RR, SRJF and ORR-SJRF

3.4 Analysis and Observations:

The **Proposed ORR-SJRF** algorithm shows the lowest average waiting time and turnaround time, outperforming both traditional RR and SJRF.

- **SJRF** performs better than traditional RR due to its pre-emptive nature, which minimizes waiting for shorter tasks. However, it can cause **starvation** for longer tasks, which ORR-SJRF avoids by retaining the cyclic fairness of Round-Robin.
- **ORR-SJRF** achieves the best balance by dynamically adjusting the time quantum and prioritizing shorter remaining burst times, leading to the most efficient scheduling among the three approaches.

4. Conclusion:

The **Proposed ORR-SJRF** algorithm provides the most optimized performance, reducing both waiting and turnaround times while enhancing CPU utilization.Its dynamic and adaptive nature makes it suitable for varied and dynamic workloads, ensuring fairness and efficiency without the starvation risk of SJRF.

References

Silberschatz, A., Galvin, P. B., & Gagne, G. (2002). *Operating system concepts*.

Alsheikhy, A., Ammar, R., &Elfouly, R. (2015). An improved dynamic Round Robin scheduling algorithm based on a variant quantum time. *Conference: 2015 11th International Computer Engineering ConferenceAt: Cairo, Egypt*. https://doi.org/10.1109/icenco.2015.7416332

Dash, A. R., Sahu, S. K., & Samantra, S. K. (2015a). An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum. *International Journal of Computer Science Engineering and Information Technology*, *5*(1), 07–26. https://doi.org/10.5121/ijcseit.2015.5102

International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, India, 2016,. (n.d.). (H. B. Parekh & S. Chaudari), *&qout;Improved Round Robin CPU scheduling algorithm: Round Robin, Shortest Job First and priority algorithm coupled to increase throughput and decrease waiting time and turnaround time*. https://doi.org/10.1109/ICGTSPICC.2016.7955294

Saxena, H., & Agarwal, P. (2012). *Introduction to CPU scheduling algorithms*. LAP Lambert Academic Publishing.