A Comprehensive Review and Comparative Analysis with Performance Insights and Simulation-Based Evaluations of Innovative CPU Scheduling Algorithms

Izzah Alam
MSCS71F24S009
University of Sargodha
izzahalamstudy@gmail.com

December 28, 2024

Abstract

Scheduling is considered a major problem in multiprogramming operating systems.CPU scheduling play a vital role in optimization of system performance. This paper aims to review classical and novel CPU scheduling algorithms to address scheduling challenges such as resource allocation, average waiting time, throughput, etc. The study evaluates CPU scheduling algorithms through some simulations and previous comparative analysis. The results shows that Round Robin and Multilevel Feedback Queue Scheduling Algorithm performs well in reducing starvation and increasing CPU performance/utilization.

Keywords: CPU Scheduling Algorithm, Round Robin, Analysis of CPU Scheduling Algorithms, Throughput, Resource Allocation

1 Introduction

Scheduling is considered the heart of computer systems because resource allocation between the processes is only decided by the scheduling technique. The process is the most basic working unit for the execution of resources that also requires resources that the CPU allocates. As the process is the smallest unit, has five states to be run. New, Ready, running, Waiting, and Terminated are the five states of a process[8].

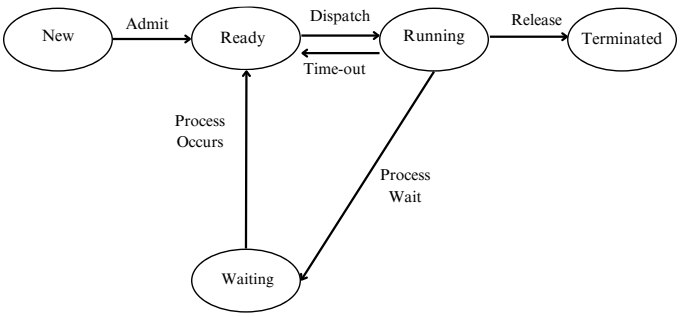

Figure 1: Five States of a Process

There are various queues where a process roams and then it aborts. The main queues are the job queue, ready queue, and I/O queue, and the whole process is carried out by schedulers.

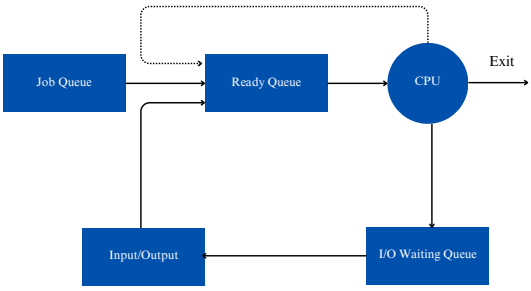

Figure 2: Different types of queues

Three types of schedulers determine which task or process is needed to migrate from queues to the CPU: long-term scheduler, short-term scheduler, and mid-term scheduler. There are two types of scheduling algorithms: preemptive and non-preemptive, which give the best CPU execution time. Scheduling involves managing resources for multiple processes and their allocation. Some of the scheduling algorithms are, First Come First Serve, Shortest Job First, Priority Scheduling, and Round Robin. Scheduling algorithms were combined to make a new algorithm called Multilevel Feedback Queue Scheduling, which reduces starvation [8]. There are some scheduling objectives such as maximum throughput, no starvation, minimal overhead, stability between response and utilization, Some scheduling criteria are, context switch, throughput, CPU utilization, Turnaround Time, Waiting Time,and Response Time [1].

The major objective is to ensure CPU utilization and decrease waiting time to improve the efficiency of the system. However, a key challenge is the management of alternating usage of the CPU and waiting time for input and output operations. It is essential to keep the CPU busy as much as possible [13].

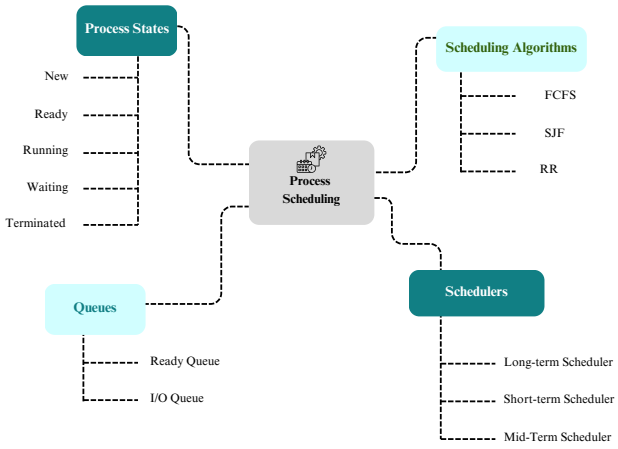

Figure 3: Key Elements of Process Scheduling

This article delves into different CPU scheduling algorithms, showing how different algorithms are better or in which way. This survey article has synthesized findings from previous studies, some comparative analyses, simulation work, and some novel work of different researchers. This article will serve as a comprehensive reference for researchers, and educators in the field.

Further sections are organized as follows, Section 2 provides an in-depth literature review of different previous research and novel CPU scheduling algorithms. Section 3 provides Limitations. Section 4 provides Suggestions. Section 5 represents the conclusion by summarizing the key findings.

2 Literature Review

In modern computing, CPU scheduling plays an important role in ensuring the efficient utilization of system resources. With time advancements in technology and other fields such as business, the demand for computer systems is increasing particularly in real-time systems, and there is a need for efficient scheduling algorithms to manage and allocate the

resources efficiently so that each process gets time and starvation does not occur as well. Moreover, the systems must guarantee fairness and maximum system performance. This literature review delves into the various CPU scheduling algorithms, focusing more on Round Robin and its variants and rest algorithms. There are also comparisons between different scheduling techniques. We have explored the existing research papers, different review articles, and some novel work by researchers and tried our best to write a review article on it. This review article will aim to provide the existing work, their strength, limitations, and practical working of these scheduling algorithms.

2.1 Comparative Analysis of Scheduling Algorithms

Hoger K. et al. performed a comparative analysis of essential CPU algorithms, FCFS, RR, SJF, and PS. They have also applied it to MATLAB. The work of researchers has been mentioned in the form of a table [8].

No.	Algorithm	Complexity	AWT	Starvation	Preemption	Advantages	Disadvantages
1	FCFS	Not Complex	Large	No	No	It is simple implement.	It offers low throughput and poor I/O overlap
2	SJF	More complex than FCFS	Smaller than FCFS	No	No	It reduces the waiting time, gives priority to I/O bound jobs	It is tough to predict next CPU burst, with starvation risk
3	LJFS	More complex than FCFS	It will depend on process size	No	No	It is simple to implement	CPU monopolization
4	LRTF	More complex than FCFS	It will depend on process size	Yes	Yes	It is efficient for engineering applications	High starvation risk
5	SRTF	More complex than FCFS	It will depend on process size	Yes	Yes	It ensures fast execution for smaller jobs	overhead of context switch and risk of starvation
6	Round Robin (RR)	Complexity will depend on size of TQ	It is more as compared to SJF	No	Yes	It solved starvation problem and fair execution of processes	It is considered poor for interactive jobs
7	Priority (Preemptive)	Complex	Smaller than FCFS	Yes	Yes	It ensures execution of high priority tasks	There is risk of starvation
8	Priority (Non-Preemptive)	It is moderate complex	Smaller than FCFS	Yes	No	It is efficient for high priority tasks.	there is inequity in process priority
9	Multilevel Queue	More complex than Priority	Smaller than FCFS	Yes	No	It is suitable for mixture of workload	Starvation risk
10	Multilevel Feedback Queue	Complexity depends on size of TQ	It is less in many cases	No	No	It prevents starvation, optimizes turnaround time.	Difficult to configure queue,

Table 1: Comparison of 10 different algorithms in terms of key performance metrics

Lalit Kishor et al. performed a simulation on MATLAB and evaluated the performance of 10 processes that arrive simultaneously under four scheduling algorithms, FCFS, SJF, RR (time quantum of 9 ms), and PS. The result is given below in the form of a table [9]

Scheduling Algorithm	CPU Engagement	Throughput	Turnaround Time	Waiting Time
FCFS	High	Low	1657 (High)	1375 (High)
SJF	Medium	High	1082 (Medium)	791 (Medium)
RR	High	Medium	1837 (High)	1555 (High)
Priority	High	Low	1473 (Medium)	1191 (Medium)

Table 2: Comparison of Scheduling Algorithms Based on Key Metrics
[9]

The Average Waiting times of the three primary algorithms discussed by Fifin Sonata et al, are RR, SJF, and FIFO. It is a metric for assessing the efficiency of Scheduling Algorithms. It represents the average amount of time a job or process stays in wait in the ready queue to execute. The researchers have followed a specific methodology to calculate AWT by Gantt Chart for SJF, FIFI, and RR. They have shown how processes are executed and demonstrated the performance of different algorithms [13]. The comparison table is here below

Metric/Algorithm	Shortest Job First (SJF)	First-In-First-Out (FIFO)	Round Robin (RR)
AWT (Average Waiting Time)	443 seconds	531 seconds	840 seconds
Scheduling Basis	Burst time	Arrival time	Time slicing (quantum-based)
Strengths	It offers low AWT and efficient for small jobs	It is simple to implement	It offers fairness in CPU allocation
Weaknesses	For longer jobs it causes starvation	It offers high AWT	High Context switching
Scenario Suitability	It is suitable when job size are predictable	It is suitable for batch processing systems	It is suitable for interactive systems
Case Study	Gantt chart: A → C → B → E → D	Gantt chart: A → B → D → C → E	Gantt chart: Alternating processes in time slices

Table 3: Comparison of CPU Scheduling Algorithms Based on AWT
[13]

El-Sharawy Enas proposed a comparative study on the CPU Scheduling Algorithms [4]. The work is here below in the form of the table:

Proposed/Reviewed Algorithms	Techniques Used	Comparison	Key Findings
FCFS, SJF, RR	Individual algorithm testing	Based on waiting time	Compared algorithms for specific processes; no single algorithm fits all cases.
Improved RR (based on k-means clustering)	k-means clustering	PWRR, TRR, PRR, SRR, ADRR	Improved performance by minimizing average waiting and turnaround time.
FCFS, SJF, PS, RR, DRR	Efficiency optimization	Compared FCFS, SJF, PS, RR, DRR	DRR perfomed better than other algorithms in terms of runtime and resource efficiency.
Hybrid RR-Priority-Based Scheduling (PB)	Combination of RR and PB techniques	Compared with traditional RR scheduling	Improved real-time performance and addressed CPU backlash in RR scheduling.

Table 4: Comparitive Analysis of Scheduling Algorithms and Techniques
[4]

Below is the comparison table only for Round Robin and its variants:

Proposed Algorithm	Techniques Used	Comparison	Key Findings
SJFDRR (Smart Job First Dynamic RR)	Dynamic time quantum	FJFDRR	It reduced context switches, AWT, and ATT for zero arrival time processes.
Improved FJFDRR	Arrival time of a process	FCFS, SJF, RR, BJF	There was balanced context switches
Enhanced RR (ERR)	Dynamic adjustments	RR, IRR	Offers lowers AWT and ATT and improves CPU utilization
ORRSM	Manhattan distance for optimal time quantum	SRR	it reduces context switches, ATT, and AWT.
HYRR	It Combines SJF, FCFS with dynamic time quantum	RR	It have Maximized CPU utilization, minimized AWT, ATT, ART, and NOC.
RR in CloudSim	Time-sharing and shared space policies	FCFS, SJF	Round Robin performed better than FCFS and SJF for cloudlet task scheduling.
Combined SJF and RR	Least burst time first	Traditional RR	It reduces RT, WT and starvation
Optimized RR	Dynamic time quantum	Traditional RR	It Increased efficiency through optimal time quantum determination.
Smart RR	Dynamic adjustment based on remaining time	Traditional RR	It lowers AWT and ART compared to RR.
RR-Priority Hybrid	Combines RR and priority scheduling	Traditional RR	It has Solved aging issues by improved fairness and efficiency.
Improved RR with Intelligent Time Quantum	Intelligent time quantum calculation	Traditional RR	It has Reduced AWT, ATT, and context switches compared to RR.
Simplified Dynamic Improved RR (STARR)	Numeric outlier detection, geometric mean	Improved RR variants	It had Lower AWT and ATT specially with outlier burst times.
Improved RR	Derived features from RR	Traditional RR	It has enhanced CPU performance and It has reducedAWT, ATT, and context switches.

Table 5: Comparison of Round Robin Algorithm and it's variants
[4]

Below is the comparison of SJF and its variants:

Proposed Algorithm	Techniques/Modifications	Comparison	Key Findings
Basic SJF	Comparison with FCFS	FCFS	SJF gives better scheduling performance than FCFS.
FCFS-SJF Priority Scheduler	Priority-based for similar priority jobs	FCFS	It has reduced average waiting and turnaround time.
SJF Implementation	Explosive time-based scheduling	FCFS	Better average waiting time compared to FCFS.
Analysis of CPU Scheduling	Comparative analysis	General scheduling	SJF increases the wait time for long processes, unsuitable for some use cases.
Multi-programming Scheduler	Based on 8 scheduling parameters	FCFS, RR, SJF	Ideal algorithm reduces response time, overheads, and increases productivity.
SJF Analysis	Optimality evaluation	General scheduling	SJF provides minimum average waiting time.
Ideal SJF Algorithm	Queue reordering for shortest burst times	None	SJF suitable for batch jobs with predefined runtimes.
New SJF Approach	Focus on reducing context switching	RR, SJF (protective)	Improved performance by minimizing context switching.

Table 6: Comparison of SJF-based Scheduling Algorithms and Techniques [4]

2.2 Novel Improvements in Scheduling Algorithms

Prem Sagar Sharma et al. did a great, research and found a novel intelligent round-robin CPU scheduling algorithm. The major performance criteria such as waiting time, response time, turnaround time, and numbers of context switching should be enhanced to get better processing. Some major scheduling algorithms are, FCFS, PS, SJF, and RR. The only thing that differentiates each algorithm from the other is, 'how the job is allocated to CPU'. The researchers mentioned the advantages and disadvantages of all the above-mentioned algorithms [2],

Algorithm	Key Characteristics	Advantages	Disadvantages
FCFS	Execution of processes depend on their arrival time	Simplest to implement	Processes with long CPU bound can delay shorter ones
SJF	Execution of processes depend on shortest burst time first	It minimizes average waiting time and turnaround time	Starvation occurs
PS	Execution of processes are based on priority values that are assigned	It ensures that critical processes are executed first	Processes with low priority can face starvation
RR	Time quantum is assigned to all processes	It ensures fair allocation of CPU time	Performance is dependent on time quantum

Table 7: Comparison of CPU Scheduling Algorithms [2]

The researchers' main focus was the Round Robin Algorithm, so they studied work before theirs, found some gaps, and proposed a new algorithm. The proposed intelligent round-robin CPU scheduling algorithm was designed to improve the average waiting time, the average turnaround time, and the context switches using the foundation RR algorithm. This algorithm was effective for real-time scheduling. [2]

The first step is process clustering. Processes are grouped into sub-ready queues (SRQs) based on their burst time using the Standard Deviation (SDJ) and optimal threshold (Pvt). Jobs with burst time SDJ less than or equal to Pvt are grouped in one SRQ; otherwise, they are distributed in different SRQs.

The next step is Time Quantum Calculation. Two quantum values are calculated for each SRQ:

The third and last step is Process Execution. Processes are executed using IQ values found in the initial SRQ. If the burst time of a process is less than or equal to the Dynamic Intelligent Quantum (DIQ), it is executed with DIQ. All processes in the remaining SRQs are executed using a minimum quantum of 1 until the current SRQ is empty. When an SRQ becomes empty, it becomes the initial SRQ.

Quantum Type	Formula	Description
IQ (Intelligent Quantum)	$IQ_i = \text{floor}\left(\frac{\text{Mean}_i + \text{Median}_i}{2}\right)$	Computed using the mean and median of burst times for the sub-ready queue (SRQ).
DIQ (Dynamic Intelligent Quantum)	$DIQ_i = IQ_i + SD_i$	Adjusted by adding the standard deviation (SD) of the SRQ to the previously computed IQ.

Table 8: Quantum Types and Formulas
[2]

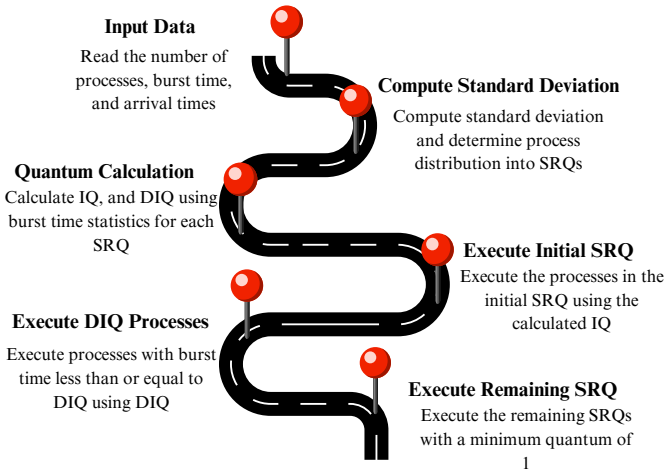

Figure 4: Working of Intelligent Round Robin Algorithm

Metric	New Intelligent Round Robin (NIRR)	Existing Algorithms (Average Values)
Standard Deviation (SDJ)	104.676433	Not Applicable
Optimal Partition Value (Pvt)	24.9	Not Applicable
Sub Ready Queues (SRQs)	4	Not Applicable
Intelligent Quantum (IQ)	Computed per SRQ (e.g., 23, 90)	Fixed Time Quantum
Dynamic Intelligent Quantum (DIQ)	Computed per SRQ (e.g., 41.04, 104.14)	Not Available
Average Turnaround Time (ATAT)	Reduced (better than others)	Higher (varies by algorithm)
Average Waiting Time (AWT)	Lower (better than others)	Higher (varies by algorithm)
Number of Context Switches (NCS)	Reduced	Higher
Response Time	Better	Worse

Table 9: Comparison of Proposed NIRR and Existing Algorithms
[2]

Tri Dharma Putra chose the Pre-emptive Shortest Job First (SJF) for analysis. In this algorithm, processes are sorted according to their burst time. It is especially suitable for batch jobs, where the runtime is known. The SJF provides the best average turnaround time for batch jobs, but starvation may occur. The researcher has discussed three case studies in this research, with arrival time and burst time [12]. Below is the comparison table:

Case Study	Theory	Process (Arrival, Burst)	Average Waiting Time (AWT)	Average Turnaround Time	Gantt Chart Timeline
1	5 processes and in preemptive nature if a process with shorter burst time arrives	P1(0,4), P2(2,3), P3(4,6), P4(7,8), P5(8,9)	4.8	10.8	0,4,7,13,21,30

2	Preemption occurs multiple times because of different burst and arrival times	P1(0,4), P2(2,6), P3(4,5), P4(7,8), P5(8,9)	6.0	12.4	0,4,9,15,23,32
3	Minimal preemption because processes have shorter burst times	P1(0,4), P2(2,3), P3(4,3), P4(7,1), P5(8,2)	1.6	4.2	0,4,7,8,10,13

Table 10: Case Studies Comparison for Pre-emptive SJF [12]

Sukumar et al. mentioned that there are different scheduling parameters such as CPU utilization, Throughput, Waiting time, turnaround time, response time, priority, and fairness. They have reviewed some previous existing CPU scheduling algorithms such as First Come First Serve, Shortest Job First, Priority Based, and Round Robin. They brought a novel algorithm that can act as preemptive and non preemptive solely based on arrival time [3]. A new factor being innovated while calculating is condition factor (F). The Burst time and arrival time of a process are added to find the condition factor.

$$F = \text{Burst Time} + \text{Arrival Time}$$

F is assigned to each process and then all processes are arranged in ascending order depending on their condition factor in the ready queue. Shortest the condition factor, Executed first. After the next minimum condition factor (F) will be executed. [3]

This algorithm has been found beneficial in reducing the waiting time, turnaround time, and response time, and the throughput and CPU utilization have been increased.

Algorithm	**Type**	**AWT (ms)**	**TAT (ms)**	**CPU utilization (%)**	**Efficiency**
FCFS	Non Preemptive	High	High	60-70%	Not efficient for diverse processes
SJF	Preemptive/ Non-Preemptive	Low	Low	80-90%	Suffers from starvation
RR	Preemptive	Medium	Medium	85-90%	Fair for time sharing
PS	Preemptive/ Non-Preemptive	It varies	It varies	60-90%	Efficiency depends on priority assignment
Proposed Algorithm	Hybrid (Preemptive and Non-Preemptive)	Lowest	Lowest	90-95%	It combines the benefits of SJF and priority with minimal starvation

Table 11:
[3]

2.3 Simulation, Survey, and Reviews

Kuldeep Vayadande et al. studied CPU scheduling algorithms having a major focus on SJF and its variants and comparison with other scheduling algorithms. below is the table. The researchers have mentioned the titles of papers with the year and their authors' names along with what algorithm they have used and their key findings. Below is the only table of used algorithms and key findings. [7]

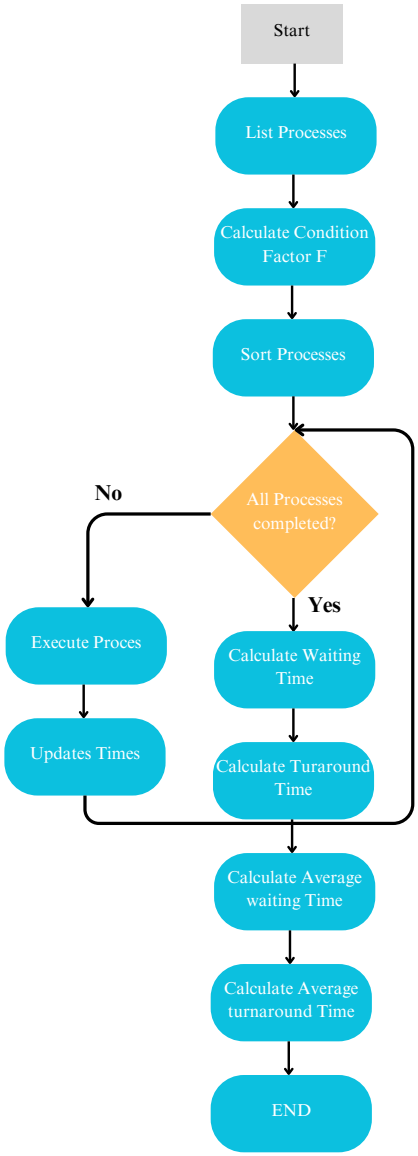

Figure 5: Flowchart for Hybrid Novel Algorithm

Algorithm Used	**Key Findings/Conclusions**
FIFO, SJF, Round Robin	Modified SJF achieved reduced context switching and improved average waiting time compared to other algorithms.
Preemptive SJF	Statistical analysis highlighted reduced waiting and turnaround times through Preemptive SJF.
Preemptive and Non-Preemptive SJF	Efficient execution in multiprocessor systems; SJF prioritized tasks with shorter execution times.
Enhanced Preemptive SJF	Addressed starvation issues and improved service quality in soft real-time systems.
Round Robin, SJF, FCFS	SJF excelled in multi-processor environments; FCFS suited for simple, short-burst processes.
FIFO, SJF	SJF improved average waiting and turnaround times significantly; suitable for concurrent process environments.
Round Robin with Dynamic Quantum, SJF	It has reducedstarvation of lengthy processes, improved turnaround, and waiting times through hybrid scheduling.

Table 12: Comparison of Algorithms and Their Key Findings

The researchers have worked on new computing innovations to check the algorithms' resource allocation and their processing time. They have found that innovations are almost better than the processors of 10 years ago and found them cheaper as well. They have tested scheduling algorithms on available processors to find out the best. They have mentioned some scheduling criteria such as efficiency, throughput, turnaround time, waiting time, response time, and fairness. Researchers have used C++ for simulation because they found it best for interfaces and it is a common base, moreover, C++ offers polymorphism. A double-ended queue was used for the waiting and completed queues. The task is assigned to the simulator randomly generating arrival time and burst time [11]. The results are here in the following.

Aspect	**First-Come, First-Served (FCFS)**	**Round Robin (RR)**	**Shortest Job First (SJF)**	**Shortest Remaining Time (SRT)**
Scheduling Type	Non-preemptive	Preemptive	Non-preemptive	Preemptive
Key Advantage	Simple and easy to implement	Fairness in time-sharing; avoids starvation	Minimizes average turnaround time	Minimizes waiting and turnaround times
Key Limitation	High average waiting and turnaround times under heavy loads	High overhead due to frequent context switching	Starvation for longer processes	Starvation for longer processes
Core Utilization	Moderate	Highest utilization across core counts	Moderate	Moderate
Average Throughput	Moderate; increases with core count	High, especially with more cores	Moderate	Moderate
Average Turnaround	Moderate; improves with more cores	Highest turnaround time for large tasks	Low	Lowest turnaround time
Average Waiting	Second highest waiting time	Highest waiting time overall	Lowest for small task counts	Lowest waiting time overall
Best Use Case	Workloads with minimal task arrival rate	Time-sharing systems; fairness-focused workloads	Systems with predictable and small task sizes	Real-time systems with frequent small tasks
Performance Trend	Performance improves as number of cores increases	Performance improves significantly with more tasks	Consistent performance across core/task counts	Consistent performance; excels with smaller tasks

Table 13: Comparison of Scheduling Algorithms: FCFS, RR, SJF, and SRT
[11]

Here below are the results from simulations

Metric	**Observation**
Core Utilization	RR consistently achieves the highest core utilization, regardless of the number of tasks or cores.
Throughput	Throughput increases with core count; RR performs better for higher core counts.
Turnaround Time	SJF and SRT outperform RR and FCFS in minimizing turnaround times, especially for larger task counts.
Waiting Time	SJF and SRT excel in minimizing waiting times, while RR and FCFS have higher waiting times as task count increases.

Table 14: Comparison of Metrics Across Scheduling Algorithms
[11]

The scheduler is considered as a 'specific system software' that is used to handle resource scheduling [10]. Naji A. Majedkan et al. proposed a review article based on various criteria such as the approach strategy and some other measures. Their work is mentioned below in the form of a table.

Proposed Scheduling Technique	Implementation Strategy	Performance Metrics	Observations & Key Contributions
Hybrid Discipline Scheduling	Combined SJF and RR techniques for effective multi-programming.	Turnaround Time, CPU Utilization	Enhanced scheduling performance; resolved existing inefficiencies.
Microscopic Traffic Control Simulation	Interfaced with AVR microcontroller for real-time traffic scheduling.	Response Time	Effective for traffic systems, real-time congestion handling.
Optimized Fixed-Priority Scheduling	Deferred preemption on uniprocessors with FNR-PA.	Turnaround Time, Waiting Time	Guaranteed optimal priority scheduling; works on multi-processor systems.
Cloud Resource Scheduling	Energy-efficient optimization for CPU, memory, network, and storage.	Energy Consumption	Optimized energy utilization in cloud environments.
Fuzzy Multilevel Queue Scheduling	Dynamic time-quantum calculation using MATLAB fuzzy toolbox.	Turnaround Time, Waiting Time	Improved queue scheduling dynamics using fuzzy logic.
Vague Logic-Based MLQ Scheduling	Prevents indefinite blocking of lower-priority tasks.	Fairness, Waiting Time	Advanced fuzzy logic techniques for better priority management.
Enhanced MLFQ Scheduling	Dynamic quantum adjustments for better CPU-bound and I/O-bound task handling.	Response Time, Throughput	Improved fairness and performance over classical MLFQ.
Simulator-Based MLFQ	Dynamic time-slice with SJF for fair CPU allocation.	Turnaround Time, Context Switching	Balanced throughput with reduced starvation.
Genetic Algorithm-Based Scheduling	Used genetic algorithm for process allocation in uniprocessors.	Turnaround Time, Waiting Time, Context Switching	Near-optimal solutions; better efficiency compared to FCFS and SJF.
Software-Defined Network Scheduling	OpenFlow protocol-based SDN for dynamic task prioritization.	Data Traffic Efficiency	Effective scheduling in networked environments.
Controlled Preemption Scheduling (cP-EDF)	Preemption control for improved earliest deadline schedulability.	Response Time	Enhanced real-time task management with reduced preemption delays.
Educational Simulation Tool	Visual Basic Application for scheduling visualization and teaching.	Learning Efficiency	Simplified teaching and learning of scheduling techniques.

Table 15: Comparison of Various Scheduling Techniques and Their Observations [10]

Here below is Performance Comparison of Key Scheduling Techniques

Technique	Turnaround Time	Waiting Time	Response Time	CPU Utilization	Fairness	Context Switching	Best Use Case
Hybrid Scheduling (SJF+RR)	Moderate	Moderate	Moderate	High	Improved	Reduced	Multi-programming OS
Optimized Fixed-Priority	Low	Low	Low	High	Moderate	Moderate	Multi-processor Systems
Fuzzy Multilevel Queue	Low	Low	Moderate	Moderate	High	Moderate	Dynamic priority-based CPU Scheduling
Genetic Algorithm	Very Low	Very Low	Moderate	High	High	Low	Uniprocessor Task Scheduling
Controlled Preemption (cP-EDF)	Moderate	Moderate	High	Moderate	Moderate	Moderate	Real-time Systems
Software-Defined Networks	High	High	Moderate	Moderate	High	Low	Network Traffic Management

Table 16: Performance Comparison of Scheduling Techniques [10]

They have mentioned some limitations such as, PS providing overhead medium CPU, low response time, medium level turnaround, and less throughput., Future work is to find an easy solution to overcome limitations

3 Suggestions for Future Work

After reviewing previous novel works, comparative analysis, surveys, and different CPU scheduling techniques, here are some future suggestions..

- here must be further work to incorporate the starvation problem, specifically in high-workload systems.
- Researchers must collaborate with industry to test new algorithms in real-world environments to get more practical benefits.

- Scheduling algorithms must be scalable to manage resources in modern computers because they deal with a large number of processes and threads

4 Conclusion

In conclusion, CPU scheduling is a basic aspect of OS design because it plays a major role in managing resources to improve efficiency and fairness in environments. This paper has explored different classical and modern CPU scheduling algorithms, mentioning their strengths, weaknesses, and their suitability for different workloads, Scheduling Algorithms such as FCFS, SJF, RR, PS, and MLFQ are discussed in this paper, in terms of important performance metrics by different researchers such as waiting time, turnaround time, CPU utilization.

No scheduling algorithm is optimal in all instances but the choice of algorithm depends on the needs and constrains of the system, dependent on available resources. FCFS and RR are widely used. Despite the progression, there are some limitations that are required to be addressed such as minimization of starvation, and improvement in fairness. To meet the growing demands of computational efficiencies, fairness, and complex computing environments there must be continuous evolution of CPU scheduling Algorithms.

Acknowledgment

The author acknowledges the guidance of Dr. Hussam Ali (email: hussam.ali@uos.edu.pk).

References

[1] I. Singh Rajput and D. Gupta, "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems ," International Journal of Innovations in Engineering and Technology (IJIET), Department of Computer Science and Engineering Amity School of Engineering and Technology, Amity University, Noida, UP, India , Oct. 2012. https://www.idc-online.com/technical_references/pdfs/information_technology/A%20Priority%20based.pdf.

[2] P. S. Sharma, S. Kumar, M. S. Gaur, and V. Jain, "A novel intelligent round robin CPU scheduling algorithm," International Journal of Information Technology, Mar. 2021, doi: 10.1007/s41870-021-00630-0.

[3] S. Bandarupalli, N. Priyanka Nutulapati, and P. Suresh Varma, "A Novel CPU Scheduling Algorithm–Preemptive & Non-Preemptive," *International Journal of Modern Engineering Research (IJMER)*, vol. 2, no. 6, pp. 4484–4490, 2012, http://www.ijmer.com/papers/Vol2_Issue6/DO2644844490.pdf.

[4] E. E. El-Sharawy, "(PDF) A Review on the CPU Scheduling Algorithms: Comparative Study," International Journal of Network Security, vol. 21, no. 1, pp. 19–26, May 2021, doi: 10.22937/IJCSNS.2021.21.1.4, https://www.researchgate.net/publication/351330519_A_Review_on_the_CPU_Scheduling_Algorithms_Comparative_Study.

[5] T. O. Omotehinwa, and J. S. Owotogbe, "A SURVEY OF VARIANTS OF ROUND ROBIN CPU SCHEDULING ALGORITHMS," FUDMA JOURNAL OF SCIENCES, vol. 4, no. 4, pp. 526–546, Jun. 2021, doi: 10.33003/fjs-2020-0404-513, urldate: 2022-01-19.

[6] M. Kumar Mishra and A. Kadir Khan, "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum," *International Journal of Computer Science, Engineering and Applications*, vol. 3, no. 6, pp. 64–69, Jun. 2012.

[7] K. Vayadande, S. Patil, S. Chauhan, R. Thakur, T. Baware, and S. Naik, "A Survey Paper on CPU Process Scheduling Keywords-CPU, scheduling, Shortest Job Next (SJN), and Shortest Remaining Time First (SRTF) (SJN)." https://www.riverpublishers.com/pdf/ebook/chapter/RP_P9788770229852C4.pdf.

[8] H. K. Omar, K. H. Jihad, and S. F. Hussein, "Comparative analysis of the essential CPU scheduling algorithms," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2742–2750, Oct. 2021, doi: 10.11591/eei.v10i5.2812.

[9] L. Kishor and D. Goyal , "Comparative Analysis of Various Scheduling Algorithms," Apr. 2013. https://www.researchgate.net/profile/Drdinesh-Goyal/publication/273260942_Comparative_Analysis_of_Various_Scheduling_A Analysis-of-Various-Scheduling-Algorithms.pdf

[10] 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, May 2020 , doi: 10.38094/jastt1215.

[11] S. Almakdi, M. Aleisa, and M. Alshehri, "Simulation and Performance Evaluation of CPU Scheduling Algorithms," *IJARCCE*, vol. 4, no. 3, pp. 1–6, Mar. 2015 , doi: 10.17148/ijarcce.2015.4301, urldate: 2019-07-27.

[12] T. D. Putra, "Analysis of Preemptive Shortest Job First (SJF) Algorithm in CPU Scheduling," *IJARCCE*, vol. 9, no. 4, pp. 41–45, Apr. 2020 , doi: 10.17148/ijarcce.2020.9408.

[13] Fifin Sonata, Juniar Hutagalung, and Aeri Rachmad, "Analysis average waiting time search performance in the queue process on CPU scheduling using the Round Robin, shortest job first and first in first out algorithm," *AIP Conference Proceedings 2679*, Jan. 2023 , doi: 10.1063/5.0111352.