A New Combination Approach to CPU Scheduling based on Priority and Round-Robin Algorithms for Assigning a Priority to a Process and Eliminating Starvation

Hussain Mohammad Abu-Dalbouh
Department of Computer Science, College of Science and Arts, Qassim University, Unaizah, Saudi Arabia

***Abstract*—The main purpose of an operating system is to control a group of processes, through a method known as CPU scheduling. The performance and efficiency of multitasking operating systems are determined by the use of a CPU scheduling algorithm. Round-robin scheduling is the best solution for time-shared systems, but it is not ideal for real-time systems as it causes more context shifts, longer wait times, and slower turnaround times. Its performance is mostly determined by the time quantum. Processes cannot have priorities set for them. Round-robin scheduling does not give more critical work greater consideration, which may affect system performance in solving processes. On the other hand, a priority algorithm can resolve processes' priority levels. This means that each process has a priority assigned to it, and processes with highest priority are executed first. If which process should come first and the process waiting time in CPU are not considered, this can cause a starvation problem. In this paper, a new CPU scheduling algorithm called the mix PI-RR algorithm was developed. The proposed algorithm is based on a combination of round-robin (RR) and priority-based (PI) scheduling algorithms for determining which tasks run and which should be waiting. The disadvantages of both round-robin and priority CPU scheduling algorithms are addressed by this novel algorithm. When using the proposed mix PI-RR algorithm, the performance measures indicated improved CPU scheduling. Other processes should not be affected by the CPU's requirements. This algorithm helps the CPU to overcome some of the problems of both algorithms.**

***Keywords*—Average turnaround time; average waiting time; utilization; performance measures; operating system; process**

I. INTRODUCTION

People can shop, learn, arrange appointments, play games, and more due to technological advancements such as mobile phones and computers. Because humans are typically unable to utilize and maintain these devices due to their complexity, operating systems have emerged to address these issues [1], [2]. They are best described as a link between the user and the computer hardware that makes managing and controlling the computer system easier. Both the user and the system benefit from the services provided by operating systems [3]. On the user side, they provide user interfaces and assist in the implementation of programs, file management, and information exchange with other computers, while on the system side, they allow multiple users to share resources and protect system resources [4], [5].

In a multitasking context, CPU scheduling is a critical duty for an operating system. A ready queue is maintained when more than one procedure needs to be executed. In a two-processor system, each processor has its own ready queue. The operating system chooses a process from a list of those in the ready queue, and assigns the CPU to it based on an algorithm [6], [7]. To ensure fairness and avoid hunger while allocating CPU to processes, close attention is essential. When making scheduling decisions, the aim is to keep the average waiting time, average turnaround time, and number of context flips as low as possible.

The operating system is in charge of managing the computer's hardware and software resources, as well as performing many functions. Processor scheduling is regarded as a fundamental task. All resources are scheduled before they are used, so they are available to processes when they are needed and at a new stage in the process life cycle [8], [9]. A short-term scheduler (STS) [10] selects a process from the ready queue for implementation, and scheduling is the essential function of the operating system in a computer system. These algorithms are used to schedule tasks in the CPU; each one outperforms the others in some performance metrics, and has its own set of benefits and drawbacks [11].

The job of a CPU scheduler is to select a process from a memory list of ready-to-run processes. In the following situations, the CPU scheduling choice for a scheduler must be made:

- Switch a process from running to ready state.
- Switch a process from waiting to ready state.
- Send a process to terminate state.

The success of the scheduler is decided by an algorithm. High-quality CPU scheduling algorithms rely on maximize usage rate, throughput, turnaround time, waiting time, and response time. In multi-processing systems, the user executes multiple applications at the same time, each of which contains multiple processes that require the CPU to complete its responsibilities, but only one process can acquire the CPU at a time. As a result, CPU scheduling is required, which allows one function to use the CPU while another waits for other resources, improving management reliability and efficiency [12]. One of the most significant components of the device is

the CPU. Because most operations rely on it, we must maximize its usage and throughput, while reducing turnaround time, waiting time, and response time. CPU scheduling techniques, which control how processes enter the CPU, can meet all of these requirements [13]. There are numerous scheduling algorithms, each of which is implemented in a unique way. The FCFS algorithm, for example, assigns the CPU to the first person who arrives. The SJF algorithm allocates the CPU to the shortest task. The round-robin algorithm assigns a time quantum to each process, calculates its working time in the CPU, then leaves the process and permits another to run. According to their priority, the priority algorithm [8] determines which processes are allowed to access the CPU. Many issues might arise during the execution of scheduling algorithms [14].

This paper proposes a mix priority and round-robin algorithm (mix PI-RR algorithm) for assigning a priority to a process and eliminating starvation. This algorithm has the optimal advantages of both priority and round-robin algorithms. The rest of this paper is organized as follows: Section II presents the literature review. In Section III, the proposed algorithm is discussed and a flowchart is presented. In Section IV, present the discussion and conclusion in the final section.

II. Literature Review

To discover the best CPU algorithm for a given procedure, it is possible to compare the three CPU algorithms based on their waiting times. Each algorithm has been extensively tested and the outcomes compared. In [15], the researchers developed an improved round-robin scheduling approach based on the clustering algorithm, which combined the advantages of prioritizing short operations with low round-robin scheduling overheads to reduce the average waiting time and turnaround time. Using the k means technique, similar processes were clustered. These researchers employed the CPU scheduling approach in [16] to create a fast system with fewer resources. They were able to improve the algorithm's efficiency and reduce its runtime. The various scheduling algorithms were designed and implemented by them. In [17], the researchers recommended scheduling techniques to increase the operating system's real-time performance.

In [18], the round-robin scheduling algorithm's time quantum concerns were addressed. The researchers developed the smart job first dynamic round-robin technique. Using a dynamic time-quantum technique, the program required the CPU schedule to sort processes in ascending order based on burst time, assign system priority, and calculate a smart priority factor (SPF) for each process. The team created a simulator to evaluate the proposed algorithm.

In [19], the researchers released an enhanced version of the Fittest Job First Dynamic Round-Robin algorithm (FJFDRR), which incorporates the process arrival time as an algorithmic element that various queues handle. The suggested approach was compared against current scheduling algorithms in four test cases using the ATAT, AWT, AR, and CS metrics. Based on the number of processes provided, the statistics demonstrated that the suggested technique had the best appropriate context switch rate.

The researchers presented the Enhanced Round-Robin (ERR) algorithm in [20], which aims to increase CPU performance by reducing the average waiting time and turnaround time. In three separate scenarios, the suggested algorithm was compared to the RR and IRR algorithms. The findings revealed that this approach performed better by reducing the average WT and average TAT.

The researchers suggested the Modified Priority Preemptive Scheduling Technique as a novel CPU scheduling algorithm in [21]. Priority pre-scheduling is implemented in a cyclical manner by the algorithm. The results indicated that the novel technique handled the starving problem, while also improving the speed of the standard preemptive algorithm.

In [22], the researchers proposed an approach for managing loads and prioritizing selection of tasks.

The SJF algorithm must be used to rank jobs, followed by the RR algorithm for execution. The findings demonstrated that the ad hoc algorithm prioritizes higher priority jobs and executes them rapidly, while contextual switching is reduced for low priority processes, reducing the options between RR and SJF.

In the Cyber-Physical System, the researcher presented a scheduling strategy for high-priority random jobs [23]. A fog group is used in idle time to process the most recent available time and execution time before assigning the system to a random high-priority task. This method speeds up the dispatch of high-priority random jobs, allowing them to be performed more quickly.

A number of CPU scheduling algorithms have been developed in recent years to ensure predictable processor allocation. Often, the best features of each algorithm have been combined to create the ideal algorithm for a given situation. The upgraded round-robin (IRR) CPU scheduling algorithm, invented by Mishra, is an improved round-robin scheduler. It is comparable to round-robin (RR), but is a little better [24]. IRR chooses the first process from the ready queue and gives it the CPU for up to one QT. When a process completes its QT, it checks the remaining CPU burst time of the presently executing process.

III. Proposed Algorithm

The round-robin algorithm does not consider the importance and significance of processes; it simply solves the queue, so the process order that the CPU performs causes a decrease in the CPU efficiency. The priority algorithm has some disadvantages because it solves the process priority for which process come first, so it performs the process with the highest priority and does not consider the arrival time or size of the process. It also ignores CPU bursts that can cause starvation. Therefore, this study created a mix of the two previous algorithms to decrease the disadvantages for the operating system environment. Mix PI-RR algorithm and flowchart of the proposed Mix PI-RR algorithm is shown in “Fig. 1 and 2”.

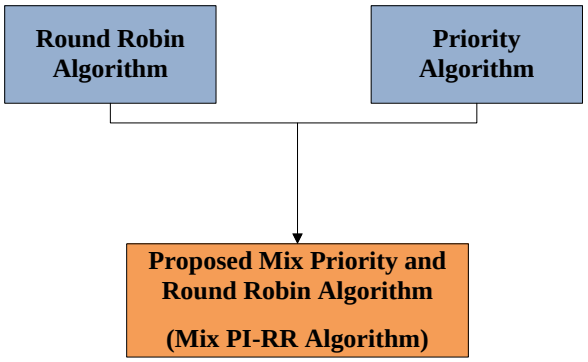

Fig. 1. Mix PI-RR Algorithm.

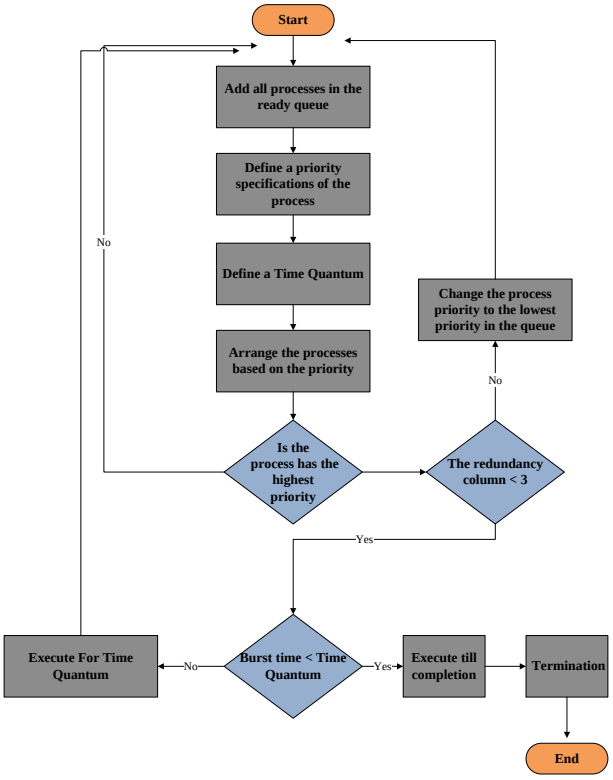

Fig. 2. Flowchart of the Proposed Mix PI-RR Algorithm.

The proposed algorithm adds two more columns to the original columns in the round-robin as follows:

Priority column: The CPU gives some processes higher priority than others. The number of times to repeat this is listed in the second column. If the procedure is prioritized three times then, in the central processing unit, the original priority is updated and the lowest priority is assigned. Following this, the processes will be carried out in accordance with the arrival time of the central processing unit (CPU). Finally, when the CPU receives the lowest priority process, the priority will be activated again. Then, it will be implemented three times before being assigned the lowest priority in the central processing unit (CPU).

NOTE 1: If more than one process had the same priority after a change, we looked at the redundancy column, and then we implemented the process with the fewest repetitions.

NOTE 2: If they were equal in the redundancy column, the arrival time to the central processing unit was taken. To break ties, processes with equal priority were completed on a FCFS basis.

NOTE 3: If a process was running and another process arrived with the highest priority, it did not interrupt the work of the process, but waited until the end of the quantum time.

IV. Discussion

Preparing a successful proposal for a newly inspired algorithm in such a pure field is not a simple undertaking. Given the difficulties of this research topic, it was necessary to propose a new optimization technique with novel aspects. Apart from the innovation, the authors' findings were supported by a dataset and a comparison of some criteria between algorithms. As a result, the three samples from the literature were utilized as examples in this study, and the average TAT and average WT of both the proposed and presented algorithms were compared. The average TAT and average WT are examined and compared to the current round-robin scheduling algorithm in different cases. Some scenarios were demonstrated, the results of each iteration were studied, and the final outputs were compared using the round-robin algorithm to verify the quality and efficiency of the suggested mix PI-RR algorithm.

Sample 1: The first dataset, which contained seven processes, was used in this sample from the benchmark datasets used in the studies. In addition to explaining how the suggested algorithm works, this research developed a Gantt chart for the method. For the following collection of processes, Table I shows the length of the CPU-burst period in milliseconds. We assigned the time quantum as 3 ms for each process. Tables II and III show gantt chart of sample 1 and turnaround time and waiting time of sample 1, respectively.

TABLE I. Processes of Sample 1

Process	CPU Burst time	Arrival time	Priority	Number of times repeat
P1	9	0	3	///
P2	9	2	2	///
P3	12	4	4	///=7
P4	8	5	1	///
P5	7	6	5	///
P6	9	6	7	///
P7	12	6	6	///=7

TABLE II. Gantt Chart of Sample 1

P1	P2	P4	P4	P4	P2	P2	P1	P1
0-3	3-6	6-9	9-12	12-14	14-17	17-20	20-23	23-26

P3	P3	P3	P5	P5	P5	P7
26-29	29-32	32-35	35-38	38-41	41-42	42-45

P7	P7	P6	P6	P6	P3	P7
45-48	48-51	51-54	54-57	57-60	60-63	63-66

TABLE III. TURNAROUND TIME AND WAITING TIME OF SAMPLE 1

Process	Turnaround Time (ms)	Waiting time
P1	26	17
P2	18	9
P3	31	19
P4	9	1
P5	36	29
P6	54	45
P7	45	33
Average	31.285	21.875

Sample 2: For the following collection of processes, Table IV shows the length of the CPU-burst period in milliseconds. We assigned the time quantum as 5 ms for each process, which contained eight processes. Tables V and VI show gantt chart of sample 2 and turnaround time and waiting time of sample 2, respectively.

TABLE IV. PROCESSES OF SAMPLE 2

Process	CPU Burst time	Arrival time	Priority	Number of times repeat
P1	15	0	4	///
P2	18	0	1, 8	/// , /
P3	15	0	2	///
P4	14	0	3	///
P5	12	0	7	///
P6	20	0	8	
P7	7	0	5	//
P8	8	0	6	//

TABLE V. GANTT CHART OF SAMPLE 2

P2	P2	P2	P3	P3	P3	P4	P4
0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40
P4	P1	P1	P1	P7	P7	P8	P8
40-44	44-49	49-54	54-59	59-64	64-66	66-71	71-74
P5	P5	P5	P2	P6	P6	P6	P6
74-79	79-84	84-86	86-89	89-94	94-99	99-104	104-109

TABLE VI. TURNAROUND TIME AND WAITING TIME OF SAMPLE 2

Process	Turnaround Time (ms)	Waiting time
P1	59	46
P2	89	71
P3	30	15
P4	44	30
P5	86	74
P6	109	89
P7	66	59
P8	74	64
Average	69.625	56

TABLE VII. PROCESSES OF SAMPLE 3

Process	CPU Burst time	Arrival time	Priority	Number of times repeat
P1	9	0	10	//
P2	12	0	9	///
P3	15	0	6	///
P4	17	0	3	///=10
P5	12	0	2	///
P6	14	0	8	///
P7	16	0	7	///,=10,/
P8	13	0	4	///
P9	6	0	5	//
P10	15	0	1	///

TABLE VIII. GANTT CHART OF SAMPLE 3

P10	P10	P10	P5	P5	P5	P4	P4
0-5	5-10	10-15	15-20	20-25	25-27	27-32	32-37
P4	P8	P8	P8	P9	P9	P3	P3
37-42	42-47	47-52	52-55	55-60	60-61	61-66	66-71
P3	P7	P7	P7	P6	P6	P6	P2
71-76	76-81	81-86	86-91	91-96	96-101	101-105	105-110
P2	P2	P1	P1	P4	P7		
110-115	115-117	117-122	122-126	126-128	128-129		

TABLE IX. TURNAROUND TIME AND WAITING TIME OF SAMPLE 3

Process	Turnaround Time (ms)	Waiting time
P1	126	117
P2	117	105
P3	76	61
P4	128	111
P5	27	15
P6	105	91
P7	129	113
P8	55	42
P9	61	55
P10	15	0
Average	83.9	71

Sample 3: For the following collection of processes, Table VII shows the length of the CPU-burst period in milliseconds. We assigned the time quantum as 5 ms for each process, which contained ten processes. Tables VIII and IX show gantt chart of sample 3 and turnaround time and waiting time of sample 3, respectively.

Scheduling is a fundamental operating system feature. Almost all computer resources are pre-programmed before they are used. One of the most important computer resources is the central processing unit (CPU). Its scheduling is crucial to the architecture of an operating system. Which processes run and which processes wait are determined by CPU scheduling. CPU

scheduling is critical because it has a significant impact on resource usage, system performance, and CPU efficiency. Process execution is made up of a cycle of CPU execution (CPU burst) and I/O wait (I/O burst), with CPU burst coming first, then I/O burst, then another I/O burst, and so on. The most recent CPU explosion ends with a system request to stop the process.

Due to their significant waiting time, long response time, large turnaround time, and low throughput, existing round-robin CPU scheduling algorithms cannot be used in real-time operating systems. Furthermore, existing priority CPU scheduling algorithms are inadequate for real-time operating systems since they create starvation, and do not take into consideration which processes come first and the time spent waiting for them to run in the CPU.

The proposed mix priority and round-robin algorithm (mix PI-RR algorithm) is an algorithm that obtains the optimal advantages of both priority and round-robin algorithms. Round-robin scheduling does not give any process priority or additional consideration based on other processes, and processes cannot have priorities set for them. Therefore, delayed execution of important processes may affect the performance of the whole system. On the other hand, in the priority algorithm, each process is assigned a priority. Processes with highest priority are executed first. However, this occurs without taking into account which process comes first and the time the process has been waiting in CPU to run, and this can cause starvation. Therefore, this paper introduced the mix PI-RR algorithm to assign a priority to important processes, without causing starvation.

The average waiting and turnaround times depend on the number of processes in the ready queue; as number of processes increases, time cost increases. In addition, long burst times of the processes increase the time cost. To emphasize the efficiency of the proposed algorithm, samples datasets varying in number and burst times of processes are used. The proposed mix PI-RR algorithm enhances CPU performance in general, the results revealed that wait time and turnaround time were reduced. Furthermore, the CPU algorithms enabled the user to obtain good results without increasing the time. Tables X and XI show the average turnaround time and average waiting time of the proposed mix PI-RR algorithm and current round-robin algorithm. As observed from the average turnaround time and average waiting time, the performance of the proposed mix PI-RR algorithm was better than the current round-robin algorithm. It is clearly observed that average turnaround time and average waiting time of the processes are optimum for proposed Mix PI-RR algorithm compared to round robin fundamental algorithm. The comparison between the proposed mix PI-RR algorithm and the current round-robin algorithm is shown in “Fig. 3 and 4”.

TABLE X. COMPARING AVERAGE TURNAROUND TIME

	Average Turnaround Time (ms)		
	Sample 1	**Sample 2**	**Sample 3**
Proposed Mix PI-RR Algorithm	**31.285**	**69.625**	**83.9**
Round Robin	**48.8571**	**90**	**106.9**

TABLE XI. COMPARING AVERAGE WAITING TIME

	Average Waiting Time (ms)		
	Sample 1	**Sample 2**	**Sample 3**
Proposed Mix PI-RR Algorithm	**21.875**	**56**	**71**
Round Robin	**39.428**	**67.375**	**94**

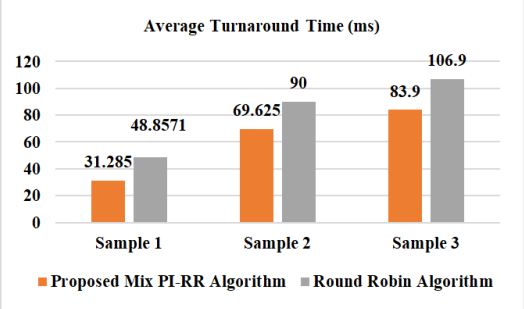

Fig. 3. Comparing Average Turnaround Time.

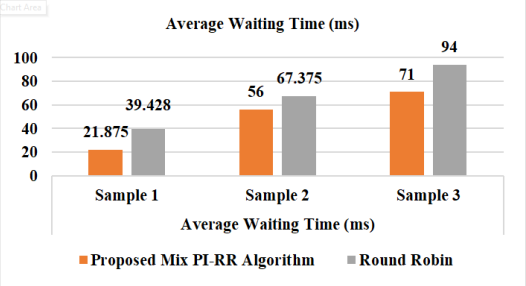

Fig. 4. Comparing Waiting Time.

V. CONCLUSION

The algorithm presented here outperforms several other algorithms; in general, it outperforms the RR and priority-based methods. No algorithm is ideal in every circumstance. It is impossible to watch a precise scheduling algorithm in action, yet precise performance can be viewed in real-time operating system operations. Several elements, such as changeable capacity, have a substantial impact on performance. This study introduced a real-time operating system and real-time tasks. We highlighted RR and priority drawbacks like high average turnaround, high context switching, high response time, high turnaround time, and low throughput, as well as the failure to take into account the process that should be first and how long processes have been waiting. After analyzing RR and priority algorithm’s performances and drawbacks, we proposed a new algorithm, named mix priority and round-robin (the mix PI-RR algorithm), which deals with the drawbacks of simple round-robin and priority algorithms. This new approach performed better than a simple RR and priority, by taking the best features of each algorithm and combining them to create the ideal algorithm for a given situation in terms of average waiting time and average turnaround time. This study justified the mix between priority and round-robin to help the CPU overcome indefinite blocking or starvation (leaving some lower priority processes waiting in CPU) in priority algorithms, and using queue up to solve the processes regardless of the importance and priority of the process for the CPU in the round-robin

algorithm. The results of this evaluation highlight ways in which instructional material should be clarified. It will be important for the proposed algorithm to be more efficient and effective than current CPU scheduling algorithms. Finally, further research is needed to compare it with other algorithms. In future work, simulations of CPU scheduling strategies are recommended. The most efficient way to evaluate a scheduling algorithm is to code it and include it in an operating system; then, the algorithm's correct working capabilities can be determined in real-time systems. Further research and studies in the future should be carried out to discover other scheduling algorithms that are optimal in certain situations and, hence, deliver the highest level of user satisfaction.

ACKNOWLEDGMENT

The researcher would like to thank Qassim University, Kingdom of Saudi Arabia. This study was supported in part by a grant from Deanship of Scientific Research, Qassim University.

REFERENCES

[1] P. B. Galvin, G.Gagne and A. Silberschatz, Operating system concepts. John Wiley & Sons. 2003.

[2] U. Shafi, M.A. Shah, A. Wahid, K. Abbasi, Q. Javaid, M. Asghar and M. Haider, A novel amended dynamic round robin scheduling algorithm for timeshared systems. Int. Arab J. Inf. Technol., 17(1), 90-98. 2020.

[3] M. Aijaz, R. Tariq, M. Ghori, S.W. Rizvi and E.F. Qazi, Efficient Round Robin Algorithm (ERRA) using the Average Burst Time. In 2019 International Conference on Information Science and Communication Technology (ICISCT) (pp. 1-5). IEEE. 2019.

[4] S. Mody and S. Mirkar, Smart Round Robin CPU Scheduling Algorithm For Operating Systems. In 2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT) (pp. 309-316). IEEE. (2019, December).

[5] H. B. Parekh and S. Chaudhari, 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. In 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC) (pp. 184-187). IEEE. (2016, December).

[6] Amit Kumar Sain, "Dynamical Modified R.R. CPU Scheduling Algorithm", International Journal of Computer Trend and Technology, Volume 4, Issue 2, PP. 90-93, ISSN:2231-2803. 2013.

[7] M.A. Alworafi, A. Dhari, A. Al-Hashmi and A.B. Darem, An improved SJF scheduling algorithm in cloud computing environment. In 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT) (pp. 208-212). IEEE. (2016, December).

[8] A. Joshi and s. Gosswami, Modified Round Robin algorithm by using Priority Scheduling. Advances in Computational Sciences and technology, 10(6), 1543-1549. 2017.

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

[10] A. Najim and Al-Tahhan, Hybrid Algorithm for CPU Scheduling by Using Dynamic Time Quantum", Future Research Journal, Al-Hadba university Collage, PP. 99-136, ISSN 1680-9300. Iraq . (April 2014),".

[11] Neelakantagouda Patil (October 2015)," A Knapsack Based CPU Process Scheduling Using Neelsack Algorithm", (IJSEAS) International Journal of Scientific Engineering and Applied Science,Volume-1, pp. 138-144, Issue-7, ISSN:2395-3470, India.

[12] William Stallings, "Operating Systems Internal and Design Principles", 5thEdition, ISBN-10: 0-13-230998, 2006.

[13] Silberschatz, A., Peterson, J. L., and Galvin, B., "Operating System Concepts", Addison Wesley, 7th Edition, ISBN-10: 0471694665, 2006.

[14] E.O. Oyetunji, A. E. Oluleye," Performance Assessment of Some CPU Scheduling Algorithms", Research Journal of Information Technology, 1(1), pp. 22-26, 2009.

[15] M Mostafa, S., & Amano, H. (2020). Dynamic Round Robin CPU Scheduling Algorithm Based on K-Means Clustering Technique. Applied Sciences, 10(15), 5134.

[16] Farooq, M. U., Shakoor, A., & Siddique, A. B. (2017, March). An efficient dynamic round robin algorithm for cpu scheduling. In 2017 International Conference on Communication, Computing and Digital Systems (CCODE) (pp. 244-248). IEEE.

[17] 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).Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 397-400). IEEE.

[18] Gupta, A. K., Yadav, N. S., & Goyal, D. (2016). Design and Performance Evaluation of Smart Job First Dynamic Round Robin (SJFDRR) Scheduling Algorithm with Smart Time Quantum. American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS), 26(4), 66-78.

[19] Manuel, J. I., Baquirin, R. B., Guevara, K. S., & Tandingan, D. (2019, February). Fittest Job First Dynamic Round Robin (FJFDRR) scheduling algorithm using dual queue and arrival time factor: a comparison. In IOP Publishing Ltd, IOP Conf. Ser.: Mater. Sci. Eng (Vol. 482, p. 012046).

[20] Khatri, J. (2016). An enhanced Round Robin CPU scheduling algorithm. IOSR Journal of Computer Engineering (IOSR-JCE), 18(4), 20-24.

[21] Chandiramani, K., Verma, R., & Sivagami, M. (2019). A Modified Priority Preemptive Algorithm for CPU Scheduling. Procedia Computer Science, 165, 363-369.

[22] Tripathi, S., Prajapati, S., & Ansari, N. A. (2017, May). Modified optimal algorithm: for load balancing in cloud computing. In 2017 International Conference on Computing, Communication and Automation (ICCCA) (pp. 116-121). IEEE.

[23] Zhang, J., Chen, C., Zheng, H. K., & Luo, Q. Y. (2019, June). A High Priority Random Task Fuzzy Scheduling Algorithm for CPS. In 2019 Chinese Control And Decision Conference (CCDC) (pp. 482-487). IEEE.

[24] Y. Berhanu, A. Alemu and M.K. Mishra, Dynamic time quantum based round robin CPU scheduling algorithm (Doctoral dissertation). 2017.