Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 5, October 2021, pp. 2742~2750
ISSN: 2302-9285, DOI: 10.11591/eei.v10i5.2812

Comparative analysis of the essential CPU scheduling algorithms

Hoger K. Omar[1], Kamal H. Jihad[2], Shalau F. Hussein[3]
[1,3]Presidency of University of Kirkuk, University of Kirkuk, Iraq
[2]College of Science, University of Kirkuk, Iraq

Article Info

Article history:

Received Jan 18, 2021
Revised Apr 30, 2021
Accepted Jul 14, 2021

Keywords:

Average waiting time
CPU scheduling
Non-preemptive
Operating systems
Preemptive

ABSTRACT

CPU scheduling algorithms have a significant function in multiprogramming operating systems. When the CPU scheduling is effective a high rate of computation could be done correctly and also the system will maintain in a stable state. As well as, CPU scheduling algorithms are the main service in the operating systems that fulfill the maximum utilization of the CPU. This paper aims to compare the characteristics of the CPU scheduling algorithms towards which one is the best algorithm for gaining a higher CPU utilization. The comparison has been done between ten scheduling algorithms with presenting different parameters, such as performance, algorithm's complexity, algorithm's problem, average waiting times, algorithm's advantages-disadvantages, allocation way, etc. The main purpose of the article is to analyze the CPU scheduler in such a way that suits the scheduling goals. However, knowing the algorithm type which is most suitable for a particular situation by showing its full properties.

Corresponding Author:

Shalau F. Hussein
Presidency of University of Kirkuk
University of Kirkuk, Kirkuk, Iraq
Email: shalaufarhad@uokirkuk.edu.iq

1. INTRODUCTION

Scheduling is a major job of an operating system (O.S) [1]. The methods of scheduling play a big role in the performance of the central processing unit (CPU) since it decides the utilization of the resources [2]. For CPU switching between multi processes, there are a lot of algorithms. The primary aim of scheduling is to check the fairness among processes in the ready queue with increasing the throughput and minimizing some undesirable things such as the average waiting time [3]. Processes in the computer system have dissimilar forms for example "dis-joint process, co-operating process, and interacting processes". Figure 1 presents the process lifecycle [4].

Relying on the process types and features, the processes will be located in the process control block (PCB). All the computation of the processes and also all of the data flow that happens inside the computer is controlled by this CPU with using one of the scheduling algorithms [5]. In the environment of multiprocessing, there is a lot of process at the same time consequently so, the demand for an efficient scheduling algorithm is very necessary [6]. The main types of CPU scheduling algorithms are: "first come first serve (FCFS), shortest job first (SJF), priority scheduling, and Round Robin (RR)" [7]. Furthermore, with the huge development in this technical era the serve of "very large-scale integrated circuit VLSI" is a good potential to create a processor with high power [8]. A Unitary standard that must be accomplished through scheduler is reducing the average waiting time for the processes entirely [9]. The operating system

usually has three scheduler types which are long-term scheduler also called high-level (long-term) scheduler, mid-term scheduler and short-term scheduler it also called CPU scheduler or dispatcher [10].

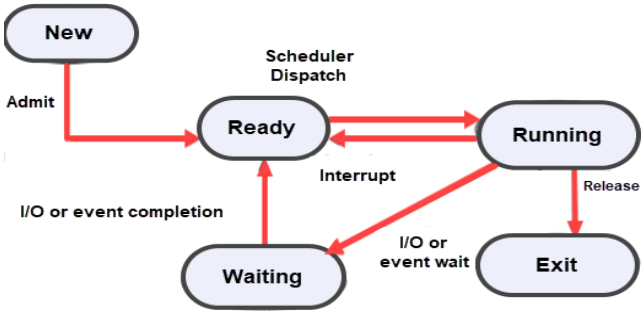

Figure 1. Process lifecycle [4]

The long term determines which tasks or processes are to be accepted to get the ready queue so when establishing an attempt for process execution, its admission to the band of executing processes currently which is either authorized or stayed (delay) through the long-term scheduler. Hence, this scheduler orders which processes are to run on a system and the concurrency level should be backed at any time.

The mid-term scheduler temporarily deletes the processes of primary memory and puts them on the secondary memory for example, put them in a disk drive or contrariwise. This is usually concerned as processes swapping out or processes swapping in as well as incorrectly as paging out or paging in [11].

The short-term scheduler which is also called the CPU scheduler determines which of the processes being in the ready queue and in memory to be executed in another term of meaning which one is inserted in the CPU. As well as, it is controlled multiprogramming degrees for knowing how many jobs are accepted. This type of schedule can be preemptive by meaning that it is capable to delete processes forcibly from the CPU when it needs to settle another process to that CPU or it can be non-preemptive in that situation the scheduler is unable to remove the processes forcibly from the CPU.

The objective and motivation of this article are to know all the characteristics of the basic CPU scheduling algorithms. However, comparing between them based on many criteria for selecting appropriate algorithm according to the job and system requirements. The organization of the rest paper is depicted as; section 2 presents the related work. In section 3, the criteria of the CPU scheduling are explained. Describing the policies of scheduling algorithms are given in section 4. In section 5, many types of CPU scheduling algorithms are illustrated. In section 6, the comparisons between many scheduling algorithms are presented. Section 7 concludes the article.

2. RELATED WORK

A huge number of researches are published on the subject of operating system scheduling algorithms recently. This paper is the extension of the previous researches that printed in this field. A lot of the research workers formulate different scheduling algorithms, these different algorithms provide different work based on the model that they used to enhance these algorithms. There have been several approaches for CPU scheduling, this part of the article concentrates on the latest contributions of CPU scheduling.

S. Almakdi and *et al.*, 2015 [12] discussed the significance of multi-programming, and how scheduling the processes by utilizing several CPU scheduling algorithms. Also simulated and assessed the algorithms in the state of more than one core. As well as they make a comparison among the performance of CPU scheduling algorithms with dissimilar parameters such as processes number. In addition, R. Guadaña and *et al.*, 2013 [13] give an elaborate search on the CPU scheduling and investigated a lot of kinds of scheduling. Also, two scheduling strategies were mentioned in their papers which are the preemptive and non-preemptive scheduling strategies. For knowing which type of algorithms are suitable for the particular CPU scheduling goals, and for that many illustrations were given in each algorithm. However, they identify how does CPU helps jobs that are established by a user via a Scheduling Algorithm. CPU fulfills every program instruction in a chronological sequence and executes the fundamental arithmetic, logical, and I/O tasks of the system when the algorithm of scheduling is utilized through the CPU to cover all processes. L. Kishor and e*t al.*, 2013 [14] have talked about scheduling and then several kinds of scheduling.

Comparing the major algorithms is also presented by applying the MATLAB program which means they are implemented practically. Through such an observational install, they put an analysis for evaluating

the function of every basic scheduling algorithm. However, N. Goel and *et al.*, 2012 [10] they explained some mechanism about the handling of shortest process in SJF scheduling that tends at the consequences increasing in the long processes waiting time so, the process with the long job never served and though that gives the smallest average waiting time and average turnaround time. The suggestion is that every type of simulation that used in the CPU scheduling contains accuracy but it is limited. For that limitation, the sole solution method for evaluation is a scheduling algorithm with coding it and then insert it into the real operating system, after that the best potentiality of working in the algorithm is assessed in real-time. Furthermore, S. Nager and *et al.*, 2017 [15] find out that FCFS is the simplest scheduling algorithm and the easiest one for implementation for that reason it is working properly with batch and interactional systems. The SJF algorithm is implemented easily in case the next CPU request length is known. R.R algorithm handle every process in an equal way and provide the same time quantum (TQ) for all process but, it is difficult to determine the (TQ) properly and fitted for time sharing. Whenever if the process has priority than the others it can schedule through priority scheduling. But the process with the lower priority has a problem of starvation.

Recently, these scheduling algorithms are mixed to construct a new model named multilevel queue scheduling. This type has starvation trouble for removing this problem it is changed into multilevel feedback queue scheduling. In addition, I. Qureshi, 2014 [16] This researcher provides a newer rating work research and used several algorithms, phases and methods about CPU scheduling in his paper, as well as he makes a comparison based on the overhead, throughput, turnaround time and response time of the CPU. Besides, a survey on many scheduling algorithms is illustrated through a group of parameters for example, the time of running, waiting and expecting are discussed. Moreover, K. ElDahshan and *et al.*, 2017 [17] present a huge comparison on RR algorithm type with focusing on the pros and cons. Also, M. Abur and *et al.*, 2011 [18] noticed after applying each scheduling algorithm via utilizing exponential distribution that RR leads to a minimum average waiting time but not in every environment. Simulations are costly and need a lot of time but, the Simulation method gives an accurate outcome in assessing the CPU Scheduling. Similarly, V. SINGH and *et al.*, 2013 [19] utilized simulation of exponential probability distribution function. This simulator is applied for computing the waiting time (W.T) since W.T is the measure for examining the performance of the CPU.

3. THE CRITERIA OF CPU SCHEDULING

Different scheduling algorithms exist and the performance of each one can be evaluated by several criteria. Also, each one has dissimilar properties, and the selection of a special algorithm may favor one processes class above another one. In selecting an algorithm to be used in a particular place there is a point which is the properties of the various algorithms. A lot of measures have been proposed for CPU scheduling algorithm comparison and, any features are applied for comparison can make a crucial difference. Some features are [20], [21]:

a. Utilization of CPU: ahead of time till CPU stays as busy as possible, or maintain the CPU 100% busy all the time.
b. Throughput: the number of processes that finish the execution per time unit.
c. Burst time: the total time needed for executing the process.
d. Completion time: when the process finishes its execution that is called the completion time.
e. Turn-around time: the needed time to execute a process which means starting from the submission time until the completion time. It is referred by:
   Turnaround time=completion time-arrival time
f. Time of waiting: the sum of time that process spent in a queue. It is referred by:
   Waiting time=turnaround time-burst time
g. Response time: the quantity of time starting from the submission of a request to producing the first reaction.
h. Fairness: ensuring that every process obtains a fair CPU share.

4. CPU SCHEDULING POLICIES

Generally, the policies of scheduling either pre-emptive or non-preemptive [22].

4.1. Non-preemptive scheduling

Non-preemptive occurs in a multiprogramming system which means the short-term scheduler permits the process to run until it terminates or in some cases waiting for an event. In another word, the current process frees the CPU either by finishing or by changing to the state of waiting.

4.2. Preemptive scheduling

Pre-emptive rules push the process which is active currently for releasing the CPU on particular events for instance, a clock interrupts, input & output interrupts and a call of the system. The process that is executed currently needs to release the CPU involuntarily if a high-priority process puts into the ready queue.

5. CPU SCHEDULING ALGORITHMS

Scheduling stands for deciding which jobs run whenever there are more than runnable jobs. There are various objectives for competing these scheduling algorithms. These objectives are the throughput and turnaround time as well as the response time [23]. In another term of meaning the scheduling of CPU is the procedure of identifying which process in the queue will allocate firstly to the CPU. Many types of well-known scheduling algorithms will be described in the next subsections.

5.1. First come first served scheduling

It is the simplest type of scheduling algorithm and according to its name, any process that comes first will be executed first which means it is a FIFO queue. Hence there are a lot of troubles linked with this type such as if the process that served firstly is too long then other processes that are shorter must wait for a long time and as consequence, this will lead to increasing the average waiting time. This trouble is also called the convoy effect. The process does not abandon the CPU until it either finishes or performs I/O [24].

5.2. Shortest job first scheduling algorithm (non-preemptive)

In this scheduling algorithm, the selection of the process is according to the smallest burst time of the process execution. Because we get a minimum waiting and turnaround time, this scheduling algorithm is good when compares with FCFS scheduling algorithms [25]. But also, it contains some weaknesses such as it is extremely hard to know the following CPU burst time request. As well as this algorithm is not applied for the shortest level CPU scheduling. Finally, the major drawback of this type is the process starvations [26].

5.3. Longest job first scheduling non-preemptive

This type of algorithm is directly the opposite applies of the SJF type. It differs from its working by taking and putting the processes that are longer jobs before the shorter jobs to the CPU. In general, the advantages consist of the easiness of calculating the longer jobs that causes calculating the shorter jobs much easier. This type of schedule is very practicable in the engineering outfit especially in electro-mechanics [27].

5.4. Longest remaining time first scheduling

In this type, the burst time (BT) of the process is determined and examined after all the time units. After finishing one unit the process that consisting of the biggest burst time will be scheduled the following. It is also known as longest job first scheduling (LJFS) preemptive type.

5.5. Shortest remaining time first scheduling

In this case of the scheduling algorithm, the ready queue is prepared based on the burst times of the processes. The procedures which need a little measure of time for completion are put ahead of the queue [28]. It is preemptive and this goes to the partition of the process into two divisions, hence producing extra context switching. And the process burst time is watched after all time units. After finishing one unit check the process that inducing the shortest burst time to be next scheduled. This type also is known as the "Shortest Job First Scheduling-Preemptive type".

5.6. Round Robin

The RR CPU scheduling algorithm is cited as standard RR and it is a preemptive type that allocates a slice of time named TQ which stands for time quantum. For every process in the ready queue [29]. Whenever the TQ completes, the current process is preempted and put in the rear of the ready queue [30]. RR is usually applied in real-time and time-sharing operating system because it provides every process an average share of time to utilize the CPU and gives a small response time. Moreover, the Standard RR algorithm has many weaknesses such as small throughput and big turnaround time as well as the big waiting time and also the huge context switches number [31]. The most exciting thing with the RR algorithm is the time quantum. On the one hand, putting a low TQ makes many context switches that lead to low performance of the CPU On the other hand, putting a large TQ might lead to a bad response time. The RR algorithm works as FCFS.

5.7. Priority scheduling for preemptive and non-preemptive types

In the priority scheduling algorithm, there is a classification of processes according to some system criteria depending on the processed type. The priority of the process is made by a group of measures and any process inserting the ready queue provides its importance as a priority. Only the priority number determines the allocating decision to the CPU for a process in a way that the high-value priority of the process will arrive at the CPU first or next. Two types of version exist for this algorithm which is preemptive and non-preemptive. In the preemptive Type of this algorithm, the lowest priority process may be suffering from starvation in case of a process with big priority keep to coming to the ready queue [32], [33].

5.8. Multilevel queue scheduling

The processes can be categorized into dissimilar parts based on their position. For instance, the general partition among processes is foreground processes and background processes. Foreground processes are known as interactive processes and background processes are known as batch processes. These cases of processes contain dissimilar response-time demands and scheduling requirements. Besides, foreground processes consist of a priority above background processes. So, the ready queue is divided into various divide queues. Also, every queue contains its scheduling algorithm. For instance, the foreground queue applies Round-Robin Scheduling plus the background queue applies FCFS scheduling type [34]. In general, the big priority process is located at the top of the ready queue stage and small priority processes are invested in a backside ready queue stage. Whenever this method is adopted then the process is put at the bottom of the ready queue stage the problem of starvation may be increased. Figure 2 illustrates the state diagram of multilevel queue scheduling (MLQ).

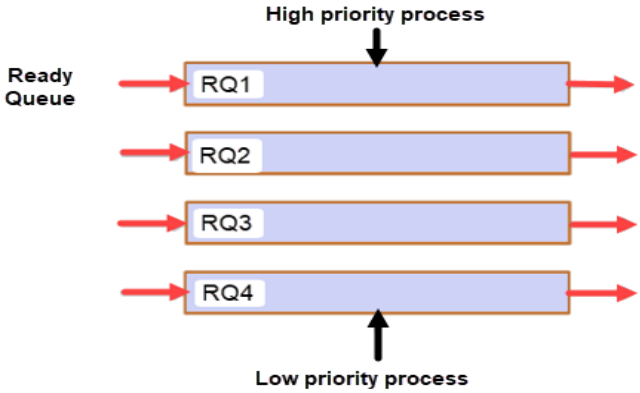

Figure 2. State diagram of multilevel queue scheduling

5.9. Multilevel feedback queue scheduling

This same idea of multilevel queue scheduling is repeated here only, the processes that do not finish their execution at the top stage are disrupted and put in the following stage of the ready queue for removing the starvation problem. But, the essential troubles with this type of scheduling are firstly: how to determine and select the optimal queue numbers for scheduling? Secondly: how much the duration of time quantum for each queue is being? Finally: how the priority is put for every job. Hence, starvation will not occur. The majority of the multilevel feedback queue scheduling (MLFQ) schedulers allow for variable quantity duration of the time quantum TQ in the queues. Usually, big priority queues are allocated to small-time quantum for interactive jobs and the smaller priority queues are allocated to the long-time quantum. Hence, there are variances in the length of TQ between several queues. This algorithm attempt to optimize the medium turnaround time and MLFQ wants that the system must be reactive more so, that it leads to reduce the response time [35]. Nevertheless, this algorithm such as RR reduces the response time but regrettably maximizes the turnaround time [36].

6. COMPARISON BETWEEN VARIOUS SCHEDULING ALGORITHMS

In this section, all the characteristics of the upper mentioned scheduling algorithms are explained by the following Tables 1-4 which are consist of many significant criteria that related to each algorithm.

Table 1. Algorithm's complexity & allocation way

No.	Algorithms	Complexity	Allocation
1	FCFS	Not complex	According to the order of the process arrives the CPU will allocate
2	SJF	More complex than FCFS	The allocation of the CPU is based on the lowest CPU burst time (BT).
3	LJFS	More complex than FCFS	The allocation of the CPU based on the highest CPU burst time (BT).
4	LRTF	More complex than FCFS	Such as LJFS the allocation of the CPU based on the highest CPU burst time (BT). But it is a preemptive type
5	SRTF	More complex than FCFS	Such as SJF the allocation of the CPU based on the lowest CPU burst time (BT). But it is a preemptive type
6	R. R	The complexity depends on TQ size	according to the order of the process arrives with fixed time quantum (TQ)
7	PR preemptive	This type is complex	According to the priority. The bigger priority task executes first
8	PR non-preemptive	This type is less complex than PR non-preemptive	According to the priority. with monitoring the new incoming higher priority jobs
9	MLQ,	More complex than the previous	According to the process that resides in the bigger queue priority
10	MFLQ	Complex but also the complexity rate depends on the TQ size	According to the process of a bigger priority queue.

Table 2. Algorithm's AWT & starvation problem

No	Algorithms	Average waiting Time (AWT)	Starvation
1	FCFS	Large.	No
2	SJF	Smaller than FCFS	Yes
3	LJFS	Depend on some measures e.g., arrival time, process size, etc.	Yes
4	LRTF	Depend on some measures e.g. arrival time, process size, etc.	Yes
5	SRTF	Depend on some measures e.g. arrival time, process size, etc.	Yes
6	R. R	Large as compared to SJF and PR	No
7	PR preemptive	Smaller than FCFS	Yes
8	PR non-preemptive	Smaller than FCFS	Yes
9	MLQ,	Smaller than FCFS	Yes
10	MFLQ	Smaller than all scheduling types in many cases	No

Table 3. Algorithm's preemption support & performance

No	Algorithms	Preemption	Performance
1	FCFS	No	Slow performance
2	SJF	No	Minimum AWT
3	LJFS	No	Big turn-around time.
4	LRTF	Yes	The preference is given to the longer jobs
5	SRTF	Yes	The preference is given to the short jobs
6	R. R	No	Each process has given a fairly fixed time
7	PR preemptive	Yes	Well performance but contain a starvation problem
8	PR non-preemptive	No	Most beneficial with batch systems
9	MLQ,	No	Good performance but contain a starvation problem
10	MFLQ	No	Good performance

Table 4. Algorithm's implementation & advantages-disadvantages

No	Algorithms	Implementation	Advantages	Dis- advantages
1	FCFS	Easiest scheduling algorithm type.	The overhead of Scheduling is short	1-Low throughput 2-This type might head to poor overlap of I/O
2	SJF	This algorithm is more difficult than FCFS. But compatible with a batch system	1-reduce waiting time 2-I/O jobs have a priority above CPU-bound jobs	Knowledge about the duration of the next CPU is needed & it is hard to estimate
3	LJFS	Easy, it is the opposite of SJF	Implementation is easy.	Monopolise CPU
5	SRTF	difficult applying it in interactional systems	The little process executes fastly.	potential of Starvation
6	R. R	In this type, the performance relies on the time quantum size	1-Performs all processes fairly. 2-It is solved the Starvation trouble.	1-Difficult to maintain TQ 2-Equal timeshare is not good thought in every case for instance, in highly interactive processes.
7	Priority preemptive	More complex than non-preemptive type.	Waiting time step by step growths for equal priority process.	Overhead of Scheduling & Starvation.
8	"Priority non-preemptive"	Moderate complexity.	Big priority jobs are completed much faster.	Get complex when priority selection of process is not fair.
9	"MLQ"	This type is hard to understand.	Various algorithms are used for various process types.	The problem of starvation.
10	"MFLQ"	More complex than MLQ & the performance relies on the TQ size.	Neglect the starvation problem	Additional context switching is wanted.

7. CONCLUSION AND FUTURE WORK

CPU scheduling is one of the extremist and significant duties of the operating system. Scheduling algorithm should be selected relying on the whole essential scheduling factors such as response time, waiting time, and turnaround time, in addition, all the other factors should be regarded as a major role for construction. In this paper, a comparison between ten major scheduling algorithm characteristics is introduced. Furthermore, a brief analysis of nine important criteria and measures that belongs to the mentioned ten algorithm types are presented. For instance, performance, implementation, advantages, and disadvantages as well as some other factors are demonstrated for showing the strength and weakness of every single algorithm type. This study concludes that there are a lot of techniques for CPU scheduling but on the other hand, none of them meets all the requirements. Because every single scheduling algorithm has its weakness. For example, FCFS and RR have a long average waiting time. However, SJF, LJFS, LRTF, and SRTF have a starvation problem, and MFLQ needs Additional context switching. So, selecting the best algorithm type depends on the job and the system type according to the mentioned algorithm characteristics in the tables. Nevertheless, there are also a lot of things that may occur in future researches for gaining the best performance with minimum cost. Further investigations may find the ideal scheduling algorithms and provide an effective solution in this field.

REFERENCES

[1] K. Noon, "A new Round Robin based scheduling algorithm for operating systems: Dynamic quantum using the mean average," *International Journal of Computer Science Issues*, vol. 8, no. 3, pp. 224-229, May 2011.

[2] R. Dash, S. K. Sahu and S. K. Samantra, "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum," *International Journal of Computer Science, Engineering and Information Technology (lJCSEIT)*, vol. 5, no. 1, pp. 7-26, February 2015, doi: 10.5121/ijcseit.2015.5102.

[3] A. Alsheikhy, R. Ammar and R. Elfouly, "An improved dynamic Round Robin scheduling algorithm based on a variant quantum time," *2015 11th International Computer Engineering Conference (ICENCO)*, 2015, pp. 98-104, doi: 10.1109/ICENCO.2015.7416332.

[4] S. Zouaoui, L. Boussaid and A. Mtibaa, "Priority based round robin (PBRR) CPU scheduling algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, p. 190-202, February 2019, doi: 10.11591/ijece.v9i1.pp190-202

[5] M. Hannats, H. Ichsan and W. Kurniawan, "CPU implementation using only logisim simulator to achieve," Bulletin of Electrical Engineering and Informatics, p. 748, April 2020.

[6] H. Kim, I. E. Hajj, J. Stratton, S. Lumetta and W. Hwu, "Locality-centric thread scheduling for bulk-synchronous programming models on CPU architectures," *2015 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 2015, pp. 257-268, doi: 10.1109/CGO.2015.7054205.

[7] J. Khatri, "An Enhanced Round Robin CPU Scheduling Algorithm," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 18, no. 4, p. 20, Jul.-Aug. 2016, doi: 10.9790/0661-1804022024.

[8] J. A. Trivedi and P. S. Sajja, "Improving efficiency of Round Robin scheduling using Neuro Fuzzy approach," International Journal of Research and Reviews in Computer Science, vol. 2, no. 2, pp. 308-311, 2011.

[9] M. U. Siregar, "A New Approach to CPU Scheduling Algorithm: Genetic round robin," *International Journal of Computer Applications*, vol. 47, no. 19, pp. 18-25, June 2012.

[10] N. Goel and R.B. Garg, "A Comparative Study of CPU Scheduling Algorithms," *International Journal of Graphics & Image Processing*, vol. 2, no. 4, pp. 245-251, November 2012.

[11] Abdulrafa Hoseen Maree, Najim Abdullah Tahhan, Maher Talal Alasaady, "New Optimized Priority CPU Scheduling Algorithm by using Knapsack (NOPSACK)," *GRD Journals- Global Research and Development Journal for Engineering*, vol. 5, no. 6, pp. 24-31, May 2020.

[12] S. Almakdi, M. Aleisa and M. Alshehri, "Simulation and Performance Evaluation of CPU Scheduling Algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 3, pp. 1-5, March 2015, doi: 10.17148/IJARCCE.2015.4301.

[13] R. R. Guadaña, M. R. Perez, L. Rutaquio Jr., "A Comprehensive Review for Central Processing Unit Scheduling Algorithm," *IJCSI International Journal of Computer Science Issues*, vol. 10, no. 1, No 2, pp. 353-358, January 2013.

[14] L. Kisho and, D. Goyal, "Comparative Analysis of Various Scheduling algorithms," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, no. 4, pp. 1488-1491, April 2013.

[15] S. K. Nager and N. S. Gill, "Comparative Study of Various CPU Scheduling Algorithm," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 6, no. 2, pp. 40-45, March - April 2017.

[16] I. Qureshi, "CPU Scheduling Algorithms: A Survey," *Int. J. Advanced Networking and Applications*, vol. 05, no. 04, pp. 1968-1973, 2014.

[17] K. El Dahshan, A. A. Elkader and N. Ghazy, "Round Robin based Scheduling Algorithms, A Comparative Study," *Automatic Control and System Engineering Journal*, vol. 17, no. 2, pp. 29-40, December 2017.

[18] M. Abur, A. Mohammed, S. Danjuma and S. Abdullahi, "A Critical Simulation of CPU Scheduling Algorithm using Exponential Distribution," IJCSI International Journal of Computer Science Issues, vol. 8, no. 6, No 2, pp. 201-205, November 2011.

[19] V. Singh and T. Gabba, "Comparative Study of Processes Scheduling Algorithms Using Simulator," International Journal of Computing and Business Research (IJCBR), vol. 4, no. 2, pp. 1-9, May 2013.
[20] M. Shoaib and M. Z. Farooqui, "A Comparative Review of CPU Scheduling Algorithms," in *Proceedings of National Conference on Recent Trends in Parallel Computing (RTPC - 2014)*, November 1-2, 2014, pp. 20-28.
[21] R. Mahadevan and N. Anbazhagan, "An Efficient Framework to Improve QoS of CSP using Enhanced Minimal Resource Optimization based Scheduling Algorithm, "*Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 3, pp. 1179-1186, December 2018, doi: 10.11591/ijeecs.v12.i3.pp1179-1186.
[22] P. K. Suri and S. Mittal, "Design of Stochastic Simulator for Analyzing the Impact of Scalability on CPU Scheduling Algorithms," *International Journal of Computer Applications*, vol. 49, no. 17, pp. 4-9, July 2012, doi: 10.5120/7717-1065.
[23] S. J. Kadhim and K. M. Al-Aubidy, "Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm," in *Communications in Computer and Information science*, vol. 70, pp. 45-52, January 2010, doi: 10.1007/978-3-642-12214-9.
[24] H. Arora, D. Arora, B. Goel and P. Jain, "An Improved CPU Scheduling Algorithm*," International Journal of Applied Information Systems (IJAIS)*, vol. 6, no. 6, pp. 7, December 2013, doi: 10.5120/ijais13-451057.
[25] S. M. Ali, R. F. Alshahrani, A. H. Hadadi, T. A. Alghamdi, F. H. Almuhsin and E. E. El-Sharawy, "A Review on the CPU Scheduling Algorithms: Comparative Study," *IJCSNS International Journal of Computer Science and Network Security*, vol. 21, no. 1, pp. 22-26, January 2021, doi: 10.22937/IJCSNS.2021.21.1.4.
[26] Xiao-Zhi Gao, S. Gurumurthy and S. Venkatesan, "Improved CPU Utilization using Advanced Fuzzy Based CPU Scheduling algorithm (AFCS*)," International Journal of Electrical Sciences & Engineering (IJESE)*, vol. 1, no. 1, p. 1-5, 2015.
[27] M. R. Mahesh Kumar, B. R. Rajendra, C. K. Niranjan and M. Sreenatha, "Prediction of length of the next CPU burst in SJF scheduling algorithm using dual simplex method," *Second International Conference on Current Trends In Engineering and Technology - ICCTET 2014*, 2014, pp. 248-252, doi: 10.1109/ICCTET.2014.6966296.
[28] M. Akhtar, B. Hamid, I. ur-Rehman, M.a Humayun, M. Hamayun and H. Khurshid, "An Optimized Shortest job first Scheduling Algorithm for CPU Scheduling," *Journal of Applied Environmental and Biological Sciences*, vol. 5, no. 12, pp. 42-46, October 2015.
[29] S. B. Bandarupalli, N. P. Nutulapati and P. S. Varma, "A novel CPU Scheduling Algorithm–Preemptive & Non Preemptive," *International Journal of Modern Engineering Research (IJMER)*, vol. 2, no. 6, pp. 4484-4490, Nov-Dec 2012.
[30] R. Roshan and K. S. Rao, "Least-Mean Difference Round Robin (LMDRR) CPU Scheduling Algorithm," *Journal of Theoretical and Applied Information Technology*, vol. 88, no. 1, pp. 51-56, June 2016.
[31] C. McGuire, and Jeonghwa Lee, "Comparisons of Improved Round Robin Algorithms*," Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, pp. 158-161, 2014.
[32] P. Singh, A. Pandey and A. Mekonnen, "Varying Response Ratio Priority: A Preemptive CPU Scheduling Algorithm (VRRP)," *Journal of Computer and Communications*, vol. 3, no. 4, pp. 40-51, doi: 10.4236/jcc.2015.34005.
[33] K. Chandiramani, R. Verma, M. Sivagami, "A Modified Priority Preemptive Algorithm for CPU Scheduling," *Procedia Computer Science*, vol. 165, pp. 363-369, 2019, doi: 10.1016/j.procs.2020.01.037
[34] J. S. Somani and P. K. Chhatwani, "Comparative Study of Different CPU Scheduling Algorithms," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 11, pp. 310-318, November 2013.
[35] S. Raheja, R. Dadhich and S Rajpal, "Designing of vague logic based multilevel feedback queue scheduler," *Egyptian Informatics Journal*, vol. 17, no. 1, pp. 125-137, September 2015, doi: 10.1016/j.eij.2015.09.003.
[36] S. Raheja, R. Dhadich and S. Rajpal, "An optimum time quantum using linguistic synthesis for round Robin CPU scheduling algorithm," *International Journal on Soft Computing (IJSC)*, vol. 3, no. 1, pp. 57–66, 2012, doi 10.5121/ijsc.2012.3105.

BIOGRAPHIES OF AUTHORS

Hoger K. Omar is currently an instructor at the University of Kirkuk and head of the Lab section in the quality assurance Department/presidency of Kirkuk University. His research interests include Big Data Analysis, Data Mining, Web Mining, Text Classification, Machine Learning, Operating systems, Distributed System with Hadoop. He received a bachelor's degree in Computer Science from the University of Kirkuk / College of Science, Kirkuk, Iraq in 2008 and a Master's degree in Information Technology from SPU University, Sulaimaniyah, Iraq in 2019.

Kamal H. Jihad is currently an instructor at the University of Kirkuk, Kirkuk, Iraq. He received a B.Sc. degree in Computer Science from Kirkuk University/ College of Science, Kirkuk, Iraq in 2007 and an M.Sc degree in Computer Engineering from Selcuk University, Konya, Turkey in 2011. His research interests include Intelligent Systems, Artificial Intelligence, and Operating systems.

Shalau F. Hussein is currently an instructor at the University of Kirkuk, Kirkuk, Iraq. She received a B.Sc. degree in Computer Science from Kirkuk University/ College of Science, Kirkuk, Iraq in 2007 and an M.Sc degree in Information Technology from Cankaya University, Ankara, Turkey in 2014. Her research interests include Operating systems, GIS, and AI.