Ingénierie des Systèmes d'Information
Vol. 29, No. 5, October, 2024, pp. 1847-1858
Journal homepage: http://iieta.org/journals/isi

A Proposed CPU Job-Scheduling Technique Based on Round Robin Method Using Dual Synchronized Time-Slices

Salam Ayad Hussein*, Emad Issa Abdul Kareem

Department of Computer Science, College of Education, Mustansiriyah University, Baghdad 10052, Iraq

Corresponding Author Email: salamayad.77@uomustansiriyah.edu.iq

https://doi.org/10.18280/isi.290517

Received: 18 December 2023
Revised: 30 August 2024
Accepted: 14 September 2024
Available online: 24 October 2024

Keywords:
operating system, task scheduling, round robin (R.R.), time quantum (T.Q), optimization techniques

ABSTRACT

A major aspect of distributed systems is task scheduling. How significant is it to adequately allocate tasks to each one of the computer's processors for the purpose of achieving better performance? The indicated approaches attempt to reduce costs and improve performance while maximizing the utilization of the central processing unit (CPU) in such a situation. The main flaw in such methods is that they require a lot of scheduling and, as a result, a lot of time. To address such a flaw, a modified CPU job-scheduling method was suggested, which used the Round Robin technique (R.R.) and dual synchronized time-slices to reduce jobs' waiting time to be executed. Therefore, the present study proves that the suggested approach has optimization efficiency with an average percentage of (12.4%–62.07%), according to an original R.R. algorithm.

1. INTRODUCTION

Task scheduling is an important field in the operating system. Task scheduling can be divided into two types: dynamic and static. Schedules are created throughout runtime in dynamic scheduling, while no knowledge regarding the task is available until it arrives [1].

It develops schedules prior to running time and has no ability to change them in static scheduling. Comparatively, all tasks should be identified ahead of time. Put differently, a static task scheduling algorithm schedules a group of tasks on processors with known communication and processing properties to improve a few performance metrics like complexity, Makespan, and CPU utilization [1]. The hybrid optimization scheduling approach is the focus of this study. The problem of timetabling was studied for many years, and various solutions were suggested. Since the 1960s, researchers have examined adequate as well as heuristic solution methods for the university and school timetabling problem [2]. When looking at the solution approaches' history, it's indicated that many solutions were suggested for such an issue. The operations research literature has been thoroughly indicated in the construction of university schedules over the last three decades [3].

In light of the prolonged execution time identified as a pivotal challenge in this undertaking, the primary objective is to ascertain an optimal sequence for the execution of jobs. The proposed methodology will consider both the time at which jobs are initiated and the time required for their completion. By balancing the entrance and execution times, the proposed approach aims to reduce the overall waiting time for jobs.

The proposed technique is based on the concept of dual synchronisation, which differs from the traditional approach of single synchronisation. In contrast to the traditional (R.R.) method, which does not take the length of the jobs into account, this technique considers the length of the jobs to be a key factor in the scheduling process. According to the proposed approach, the expected advantages Combine single and dual synchronisation for short and long jobs respectively. This approach will facilitate the completion of lengthy jobs without impacting the execution time of shorter jobs [4].

Thus, the objective of this paper will be to minimize execution time to achieve better usage of processors. Where, our contributions will be through creating a new, hybrid scheduling technique (by taking features from basic operating system scheduling techniques) and re-arrange it in a way that ensures better performance and the highest score in its fitness function [5].

2. RELATED WORK

Hussein et al. [6] demonstrated that the suggested hybrid planning framework is more effective and exacerbates the current planning gap. It aids in the study of the operating system and the use of a multi-programming environment. Creating a single priority activity policy resulted in identical scalability testing. The value of mixing complicated real-time and soft-time jobs in the system is also assessed.

Suppiah et al. [7] presented a novel queue-based multi-level CPU scheduling system. The suggested method assigns CPU service time and dynamically decides the time quantity (TQ) between different queues. The fuzzy toolbox is used to build the programming method utilized in the Math Lab application.

In the multi-level feedback queue programming technique, Jain and Jain [8], conducted a simulation study known as Markov chain analysis to examine the influence of wait state on overall system efficiency and performance. This study also

underlines the fact that the comparison analysis is calculated in an arithmetic model with shifting values and d.

Maktum et al. [9] proposed utilizing a genetic algorithm to develop near-optimal solutions to the CPU scheduling problem. They developed a simple scheduling algorithm for uni-processor scheduling based on their evolutionary technique and compared it to SJF and FCFS scheduling to reduce average wait times.

Jawad [10] developed a neural fuzzy programming technique for the application, as well as a modified CPU scheduling algorithm, in order to increase reaction time while minimizing average time and response time. It accomplishes this by combining established schedules, such as SJF preemptive scheduling and Nero fuzzy scheduling.

Saroj et al. [11] offered a strategy that was simulated and implemented in C++ programming. This solution addresses a number of difficulties, including longer average wait times, rotation times, infinite hunger blocking, and hands-on execution. With two types of split times, the proposed timetable solved numerous difficulties and was simple to implement.

Ahmed and Brifcani [12] to learn advanced features, they employed a multi-layer artificial neural network, similar to deep neural networks. Deep learning was performed using the essential characteristics derived from the data.

Elmougy et al. [13] proposed SRDQ, a revolutionary hybrid task scheduling technique that combines Shortest Job First (SJF) and Round Robin (RR) schedulers while taking the quantity of dynamically changing jobs into consideration. The suggested algorithms are primarily based on two essential keys: the first contains a large number of dynamic jobs to balance the waiting time between short and long tasks, and the second divides the ready queue into two sub-queues, Q1 for short tasks and Q2 for lengthy tasks. and another for quick jobs. the lengths. With two Q1 tasks and one Q2 task, task assignment to Q1 or Q2 resources is mutual.

Sai et al. [14] presented a hybrid approach to solve the scheduling problem. This approach combines the shortest and longest job scheduling techniques and indicates if it is suitable for both unit-processor and multiprocessor systems depending on the results. This technique can lead to efficient solutions when additional characteristics such as waiting time and execution time are taken into account.

Himthani et al. [15] suggested a multitasking scheduling approach that enhances system throughput and decreases context switching cost. Because it includes the benefits of both systems, the suggested strategy is a hybrid of Round Robin and Shortest Remaining Time First. The quantum time update is determined by the slice bit and the remaining time of the packet in this technique.

3. CPU SCHEDULING

Processor scheduling can be defined as Allocating CPU time slots to processes procedure. In a multi-processing system, which can be regarded as a dynamic operation that grows more challenging [16, 17].

The major scheduling aims can be summarized as [18, 19]:

1. Being fair between processes.
2. Maximizing the number of completed processes for each unit time (throughput).
3. Avoiding the indefinite postponement regarding any process (Starvation Free).
4. Minimizing overhead, i.e., minimizing the wasted time in scheduling.
5. Balancing resource usage, i.e., using all resources all the time.
6. Enforcing a priority scheme for allowing certain processes to get more CPU time.
7. Degrading under heavy loads.

With regard to significance, there are numerous scheduling types which were delivered as follows [20, 21]:

3.1 First in first out scheduling (FIFO)

Processes are dispatched based on the arrival time on the Ready Queue in such a type. As soon as the process has access to the CPU, it will run to the point where it's finished. Due to the fact that FIFO scheduling isn't non-preemptive, it might be used in a single programming environment. It can't be utilized as a master scheme in multi-programming yet, only as part of its [22, 23].

Disadvantages [24, 25]:

a. Less execution time process endures, i.e., the waiting time is often excessively long.

b. Favours CPU-bound processes over I/O-bound processes.

c. The first process will get the CPU first; remaining processes can use the CPU exclusively after the existing process has achieved its execution.

3.2 Round robin scheduling

Round-robin scheduling has been considered comparable to FCFS scheduling, with the exception that the CPU bursts are given time quantum limits (T.Q.). In addition, a timer has been set to whatever value was set for time quantum in a case where a process is given the CPU. In the case where the process completes its burst prior to the expiration of the quantum timer, it's swapped out of the CPU in the same way that the standard FCFS algorithm is. The process has been swapped out of the CPU and then moved to the back end of the ready queue in a case where the timer goes off first. The ready queue is kept as a circular queue, which means that as soon as all processes have had their turn, the scheduler will give the first process another turn, etc. [24, 26].

Disadvantages [18, 25]:

a. Setting the quantum too short increases the overhead and lowers the CPU efficiency, but setting it too long may cause a poor response to short processes.

b. The average waiting time under the RR policy is often long.

c. If the time quantum is very high, then RR degrades to FCFS.

3.3 Shortest job first (SJF)

It has been indicated that an extremely simple method solves the problem; actually, it's an idea that has been borrowed from operations research [C-54, PV-56] and utilized for job scheduling in computer systems. SJF is the name of this novel scheduling discipline, and it must be easy to remember since it accurately describes policy: it runs the shortest job first, after that the next shortest, etc. [27].

Disadvantages [13, 25]:

a. SJF may cause starvation if shorter processes keep coming. This problem is solved by aging.

b. It cannot be implemented at the level of short-term CPU scheduling.

3.4 Priority scheduling

Priority scheduling has been considered a more general version of SJF, where every one of the jobs is given a priority and the maximum priority job is served first. SJF prioritizes according to the inverse of the expected burst time; the smaller the expected burst, the higher the priority. Actually, priorities are conducted with the use of integers within a fixed range, yet there isn't consensus on whether high priorities should be conducted with small or large numbers. In this study, high priorities are represented by a low number, with 0 being the highest possible priority [28, 29].

Disadvantages [30, 31]:

a. If high-priority processes use up a lot of CPU time, lower-priority processes may starve and be postponed indefinitely. The situation where a process never gets scheduled to run is called starvation.

b. Another problem is deciding which process gets which priority level assigned to it.

The major effect of the previously mentioned disadvantages is that they might cause an oscillation in the optimization efficiency of these approaches.

4. THE PROPOSED APPROACH

The suggested approach has considered a finite number of jobs, each with a different execution time (length). Those jobs have been placed in the job pool from which they have been assigned to be implemented by an OS, as shown in Figure 1 below:

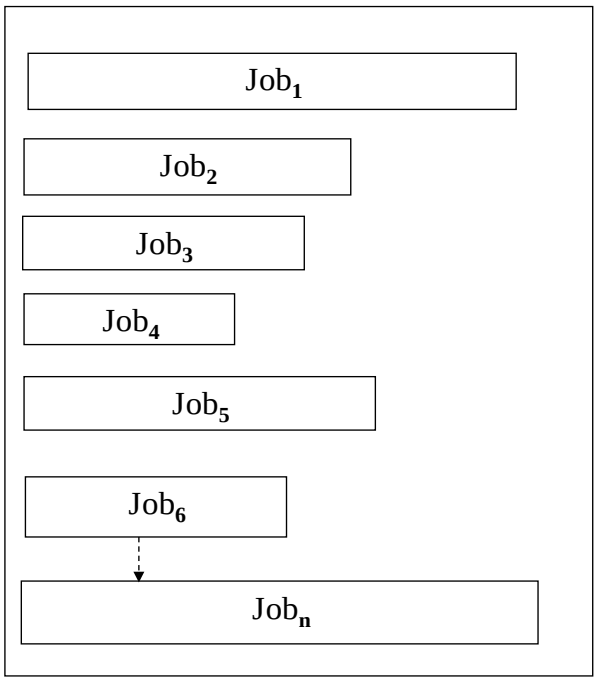

Figure 1. The proposed job's sequence

4.2 The proposed method flowchart

The suggested model will be concentrated on the job's execution time using a proposed scheduling technique via taking advantage of the R.R. technique, as in the following flowchart (Figure 2).

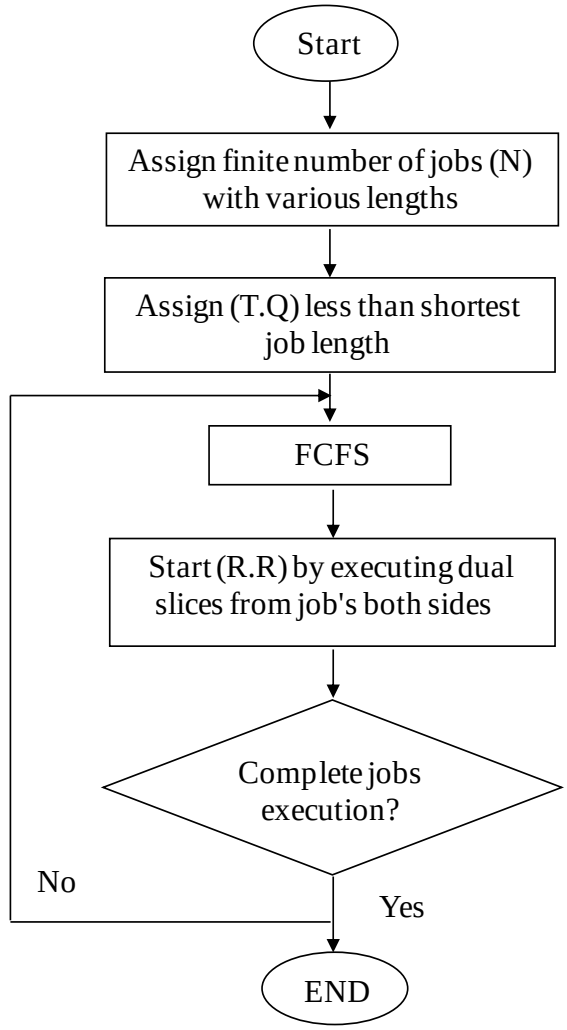

Figure 2. The proposed method flowchart

4.3 The proposed method steps

Basically, a number of jobs will be assigned (N) with various lengths. As in the R.R. technique, a time quantum (T.Q.) will be specified in order to be utilized to execute job slices from main jobs. Normally, the T.Q. should be commensurate with the length of the shortest job (T.Q. less than the shortest job length). The execution starts (based on the job's arrival sequence) by executing dual synchronized slices from both sides of each job.

For each round, three parameters will be considered as input: No. of jobs, T.Q, and length of jobs. Initially, two variables (right_side, left_side) need to be assigned. The right_side will be equal to the longest job, while the left_side needs to be assigned to be equal to the start state (initially equal to 0). For each execution round, these two variables in addition to the max_length variable will be updated using Eqs. (1)-(3) respectively (see Figure 3).

$$Right\_Side_i = Right\_Side_{\,i-1} - T.Q \quad (1)$$

$$Left\_side_i = left\_side_{i-1} + T.Q \quad (2)$$

$$Max\_length_i = Max\_length_{i-1} - T.Q \quad (3)$$

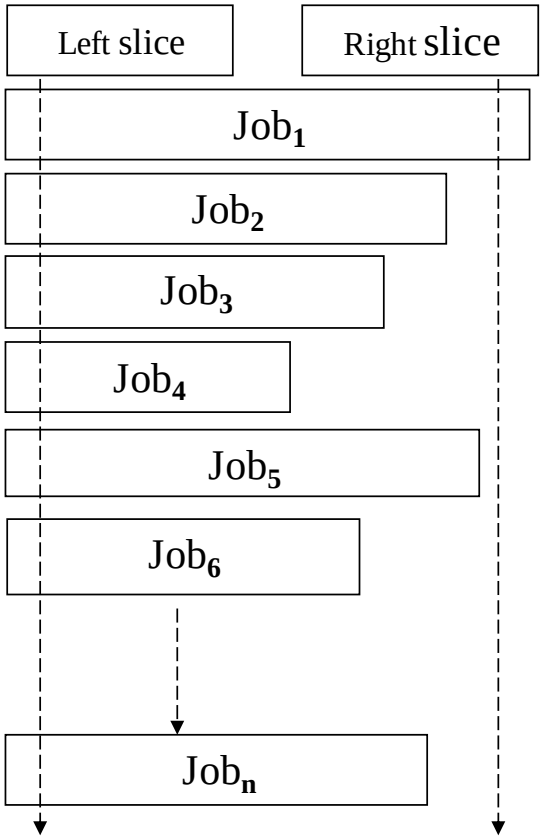

Figure 3. The proposed technique implementation

5. RESULTS AND ANALYSIS

The results have been presented via quantitative evaluation. Three hundred random jobs have been used in this issue (see Figure 4 and appendix A). The randomness came from assigning a random time for the job to enter and a random time for the job to be executed. The evaluation of the suggested scheduling algorithm has been detected by finding the optimal ratio gained via the proposed algorithm. The suggested algorithm is an optimization algorithm that balances between the two above- noted criteria (i.e., the time of job entry and the time that the job will take to be executed). The results have been presented via quantitative evaluation. The optimization ratio of the proposed approach comparing with traditional R.R has been calculated via Eq. (4).

$$optimization\_Ratio = \left(\frac{right\_side}{Job\_lingth}\right) \times 100 \quad (4)$$

In Eq. (4), the left-side variable has been ignored because it is a common factor for both approaches (i.e., traditional and proposed approaches), while the right-side factor corresponds to the proposed approach. Thus, the right-side factor contributes to the optimality of the proposed approach compared to traditional R.R.

The jobs' optimization rate will be divided into three periods: period 1 includes jobs with an optimization rate in the range (0% to 33%); period 2 includes jobs with an optimization rate in the range (33% to 66%); and period 3 includes jobs with an optimization rate in the range (66% to 100%).

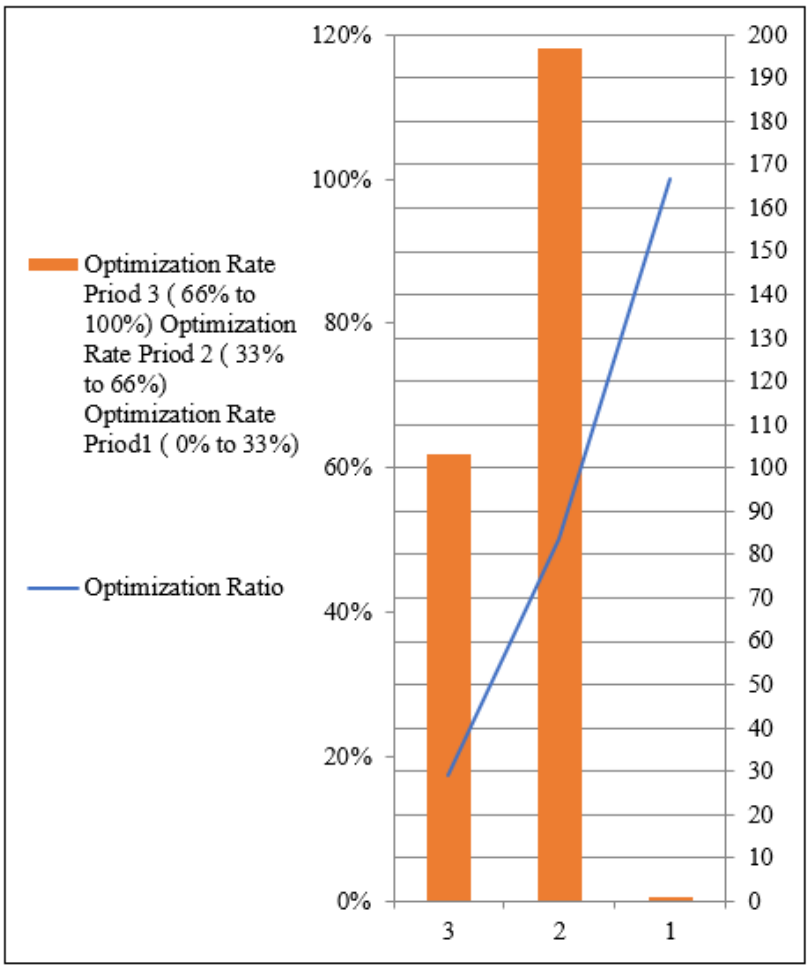

Figure 4. Optimization ratio for each job of the 300 jobs

The benefit of the proposed approach is that it does not affect the execution times of the shortest jobs, as shown in Figure 4, where jobs with an execution time that is less than half that of the longest job within the synchronized jobs group would take their own execution time. The jobs' optimization rate will improve as soon as their execution times increase. Thus, it is evident that the rate of optimization and job execution time are directly correlated. additionally, it was discovered that a job group will achieve the highest optimization rate when it is larger than 40 seconds and includes equal values, or at the very least, nearly equivalent values.

Finally, it should be mentioned that the system overhead will never increase because both long and short jobs are executed at each clock cycle at least from the left side (unlike the standard R.R. that only at most executes one side of long and short jobs at each clock cycle). And for fairness and scientific integrity, it should be mentioned that the mathematical complexity might increase in this approach due to the ongoing updating in job's length measurement from the start until the end of the execution.

6. QUANTITATIVE COMPARISON

The standard round robin and the proposed method have been compared quantitatively using a sample of 400 randomly chosen jobs with random different lengths (which makes the proposed approach applicable at any different scenario). In this experiment, the attributes were: job length, start state, right-side and left side. In addition to T.Q which is less than the shortest job. When These attributes had been generated randomly. The quantitative results of the experiment are shown in Tables 1 and 2. In comparison to the conventional Round Robin, the proposed method has been able to minify the long job Time of execution to around (12.4%–62.7%), as shown in Figure 5.

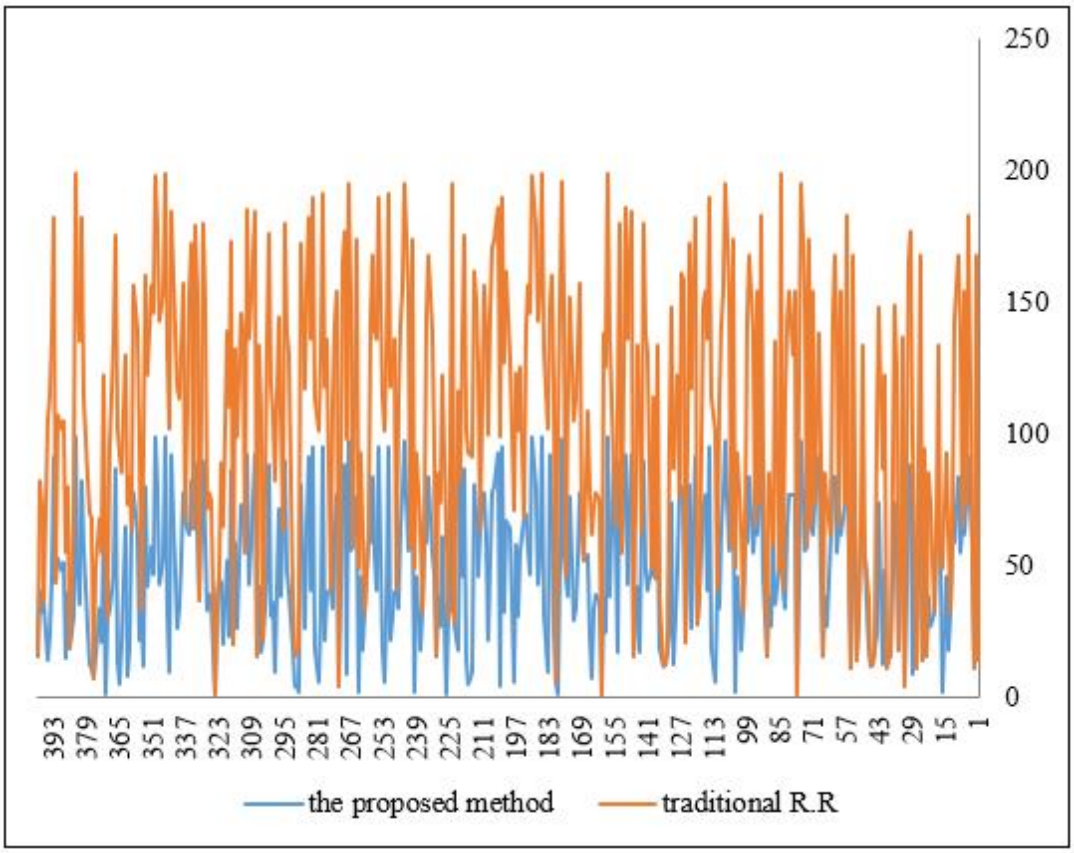

Figure 5. Quantitative comparison between traditional round robin and the proposed method

7. CONCLUSIONS

The suggested method has the priority of the execution of jobs according to entering time as well as the strategies for taking the long execution jobs into account. It combines the benefits of the round-robin method with dual synchronized time slices. In light of this, there was a noticeable decrease in the average waiting time by (12.4% to 62.07%). This proves that the proposed method is successful in reducing job execution time. This led us to the conclusion that the proposed method accomplished the primary study objective for which it was designed.

8. RECOMMENDATIONS FOR FUTURE WORK

(1) In order to shorten the time required to complete jobs, it is suggested that the classic round robin be coupled with the best aspects of spatial or temporal notions.

(2) For real time system, we recommend reducing the number of jobs or using a parallel processing to achieve the optimal results from the proposed method.

(3) In cloud computing, the proposed method can be a useful solution for task scheduling in virtual machines queues or in the request schedule manager after fog computing layer.

(4) It is recommended that the technical complexity should

be measured between classic R.R and this approach to find the system overhead.

REFERENCES

[1] Hasoon, J.N., Hassan, R.F. (2019). Solving job scheduling problem using fireworks algorithm. Journal of Al-Qadisiyah for Computer Science and Mathematics, 11(2): 1–8. https://doi.org/10.29304/jqcm.2019.11.2.557

[2] Karuppusamy, S., Muthaiyan, M. (2016). An efficient placement algorithm for data replication and to improve system availability in cloud environment. International Journal of Intelligent Engineering and Systems, 9(4): 88-97. https://doi.org/10.22266/ijies2016.1231.10

[3] Balagoni, Y., Rao, R. (2018). SAGS: A SLA-Aware green scheduling in heterogeneous cloud using hadoop YARN. International Journal of Intelligent Engineering and Systems, 11(6): 108–117. https://doi.org/10.22266/ijies2018.1231.11

[4] Banday, M.T., Khan, M. (2014). A study of recent advances in cache memories. 2014 International Conference on Contemporary Computing and Informatics (IC3I), Mysore, India, pp. 398–403. https://doi.org/10.1109/IC3I.2014.7019786

[5] Kareem, E.I.A., Hussein, S.A. (2022). Optimal CPU jobs scheduling method based on simulated annealing algorithm. Iraqi Journal of Science, 6(8): 3640–3651. https://doi.org/10.24996/ijs.2022.63.8.38

[6] Hussein, S.A., Kareem, M.R., George, D.N. (2024). Minimizing the cache memory miss ratio using modified replacement algorithm (M-CAR). Ingénierie des Systèmes d'Information, 29(2): 697-703. https://doi.org/10.18280/isi.290231

[7] Suppiah, Y., Bhuvaneswari, T., Yee, P.S., Yue, N.W., Horng, C.M. (2022). Scheduling single machine problem to minimize completion time. TEM Journal - Technology, Education, Management, Informatics, 11(2): 552-556. https://doi.org/10.18421/TEM112-08

[8] Jain, S., Jain, S. (2016). Analysis of multi level feedback queue scheduling using markov chain model with data model approach. International Journal of Advanced Networking and Applications, 7(6): 2915-2924.

[9] Maktum, T.A., Dhumal, R.A. Ragha, L. (2014). A genetic approach for processor scheduling. In International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014), Jaipur, India, pp. 1–4. https://doi.org/10.1109/ICRAIE.2014.6909108

[10] Jawad, S. (2014). Design and evaluation of a neurofuzzy CPU scheduling algorithm. In Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control, Miami, FL, USA, pp. 445–450. https://doi.org/10.1109/ICNSC.2014.6819667

[11] Saroj, S.K., Sharma, A.K., Chauhan, S.K. (2016). A novel CPU scheduling with variable time quantum based on mean difference of burst time. In 2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, pp. 1342–1347). https://doi.org/10.1109/CCAA.2016.7813986

[12] Ahmed, O., Brifcani, A. (2019). Gene expression classification based on deep learning. In 2019 4th Scientific International Conference Najaf (SICN), Al-Najef, Iraq, pp. 145–149. https://doi.org/10.1109/SICN47020.2019.9019357

[13] Elmougy, S., Sarhan, S., Joundy, M. (2017). A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. Journal of Cloud Computing, 6: 12. https://doi.org/10.1186/s13677-017-0085-0

[14] Sai, R.V., Keerthana, S., Madhiarasi, S., Preethi, S.A (2019). Competent approach to predict dengue diseases using a hybrid approach in machine learning algorithm. International Journal of Engineering Research & Technology, 7(1): 1–4.

[15] Himthani, P., Mishra, N.K., Pare, T., Dubey, G.P. (2021). Hybrid multi-tasking scheduling scheme based on dynamic time quantum using slice bit for improving CPU throughput. In Proceedings of the 3rd International Conference on Communication & Information Processing (ICCIP). http://doi.org/10.2139/ssrn.3916291

[16] Alazawi, S.A., Salam, M.N.A. (2021). Evaluation of LMT and DNN algorithms in software defect prediction for open-source software. In Research in Intelligent and Computing in Engineering, Singapore, pp. 189–203. https://doi.org/10.1007/978-981-15-7527-3_19

[17] Pappas, C., Moschos, T., Alexoudi, T., Vagionas, C., Pleros, N. (2022). Caching with light: First demonstration of an optical cache memory prototype. Optical Fiber Communication Conference (OFC), pp. Th4B.3. https://doi.org/10.1364/OFC.2022.Th4B.3

[18] Aalsaud, A., Rafiev, A., Xia, F., Shafik, R., Yakovlev, A. (2018). Model-free runtime management of concurrent workloads for energy-efficient many-core heterogeneous systems. 2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Platja d'Aro, Spain, pp. 206–213. https://doi.org/10.1109/PATMOS.2018.8464142

[19] Jaber, S., Ali, Y., Ibrahim, N. (2022). An automated task scheduling model using a multi-objective improved cuckoo optimization algorithm. International Journal of Intelligent Engineering and Systems, 15(1): 295–304. https://doi.org/10.22266/ ijies2022.0228.27

[20] Al-Jawaherry, M.A., Abdulmajeed, A.A., Tawfeeq, T.M. (2022). Developing a heuristic algorithm to solve uncertainty problem of resource allocation in a software project scheduling. Iraqi Journal of Science, 63(5): 2211–2229. https://doi.org/10.24996/ijs.2022.63.5.34

[21] Abdulmohsin, H.A., Hasan, S.S., Setapa, S. (2016). The impact of operating system on bandwidth in open VPN technology. Baghdad Science Journal, 13(1): 0204. https://doi.org/10.21123/bsj.2016.13.1.0204

[22] Kashani, M.H., Jahanshahi, M. (2009). Using simulated annealing for task scheduling in distributed systems. In 2009 International conference on computational intelligence, modelling and simulation, Brno, Czech Republic, pp. 265-269. https://doi.org/10.1109/cssim.2009.36

[23] Jahanshahi, M., Gholipour, M., Kordafshari, M.S., Rahmani, A.M. (2009). A novel method for task scheduling in distributed systems using memetic. In 2009 Second International Conference on Communication Theory, Reliability, and Quality of Service, Colmar, France, pp. 58-62. https://doi.org/10.1109/ctrq.2009.29

[24] Chen, G., Bruno, R., Salama, M. (1991). Optimal placement of active/passive members in truss structures

using simulated annealing. AIAA Journal, 29(8): 1327–1334. https://doi.org/10.2514/3.10739
[25] Advantages and Disadvantages of Various CPU Scheduling Algorithms. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-various-cpu-scheduling-algorithms/, accessed on Nov. 12, 2023.
[26] Mohan, J., Lanka, K., Rao, A.R. (2019). A review of dynamic job shop scheduling techniques. Procedia Manufacturing, 30: 34–39. https://doi.org/10.1016/j.promfg.2019.02.006
[27] Liaqait, R.A., Hamid, S., Warsi, S.S., Khalid, A. (2021). A critical analysis of job shop scheduling in context of industry 4.0. Sustainability, 13(14): 7684. https://doi.org/10.3390/su13147684
[28] Abduljabbar, I.A., Abdullah, S.M. (2022). An evolutionary algorithm for solving academic courses timetable scheduling problem. Baghdad Science Journal, 19(2): 0399. https://doi.org/10.21123/bsj.2022.19.2.0399
[29] Jain, A., Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research, 113(2): 390–434. https://doi.org/10.1016/s0377-2217(98)00113-1
[30] Cheng, R., Gen, M., Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. Computers & Industrial Engineering, 30(4): 983–997. https://doi.org/10.1016/0360-8352(96)00047-2
[31] Nasr, N.A., Elsayed, E.M. (1990). Job shop scheduling with alternative machines. International Journal of Production Research, 28(9): 1595–1609. https://doi.org/10.1080/00207549008942818

Appendix A

This appendix contains a table (Table 1) of the results of Three hundred random jobs have been used. The optimization ratio of the proposed approach comparing with traditional R.R has been calculated via Eq. (4).

Table 1. Optimization ratio for each job of the 300 jobs

NO. Job	Length	Leftt_slice	Right_slice	Optimization Rate
1	99	98	1	1.01010101
2	62	61	1	1.612903226
3	62	61	1	1.612903226
4	99	95	4	4.04040404
5	49	47	2	4.081632653
6	49	47	2	4.081632653
7	49	47	2	4.081632653
8	49	47	2	4.081632653
9	93	88	5	5.376344086
10	93	88	5	5.376344086
11	101	95	6	5.940594059
12	101	95	6	5.940594059
13	101	95	6	5.940594059
14	71	65	6	8.450704225
15	98	89	9	9.183673469
16	98	89	9	9.183673469
17	102	92	10	9.803921569
18	102	92	10	9.803921569
19	20	18	2	10
20	73	65	8	10.95890411
21	91	81	10	10.98901099
22	62	55	7	11.29032258
23	82	72	10	12.19512195
24	92	80	12	13.04347826
25	105	91	14	13.33333333
26	102	88	14	13.7254902
27	102	88	14	13.7254902
28	87	74	13	14.94252874
29	87	74	13	14.94252874
30	116	98	18	15.51724138
31	109	92	17	15.59633028
32	109	92	17	15.59633028
33	107	90	17	15.88785047
34	68	57	11	16.17647059
35	114	95	19	16.66666667
36	114	95	19	16.66666667
37	114	95	19	16.66666667
38	70	57	13	18.57142857
39	118	96	22	18.6440678
40	118	96	22	18.6440678
41	125	100	25	20
42	114	91	23	20.1754386
43	110	87	23	20.90909091
44	100	78	22	22
45	100	78	22	22
46	118	92	26	22.03389831
47	118	92	26	22.03389831
48	117	91	26	22.22222222
49	117	91	26	22.22222222
50	84	65	19	22.61904762
51	17	13	4	23.52941176
52	85	65	20	23.52941176
53	132	100	32	24.24242424
54	132	100	32	24.24242424
55	98	74	24	24.48979592
56	98	74	24	24.48979592
57	128	96	32	25
58	128	96	32	25
59	16	12	4	25
60	127	95	32	25.19685039
61	135	100	35	25.92592593
62	135	100	35	25.92592593
63	119	88	31	26.05042017
64	99	73	26	26.26262626
65	55	40	15	27.27272727
66	138	100	38	27.53623188
67	138	100	38	27.53623188
68	105	76	29	27.61904762
69	65	47	18	27.69230769
70	65	47	18	27.69230769
71	65	47	18	27.69230769
72	65	47	18	27.69230769
73	136	96	40	29.41176471
74	136	96	40	29.41176471
75	143	100	43	30.06993007
76	143	100	43	30.06993007
77	113	79	34	30.08849558
78	113	79	34	30.08849558
79	136	95	41	30.14705882
80	136	95	41	30.14705882
81	136	95	41	30.14705882
82	79	55	24	30.37974684
83	49	34	15	30.6122449
84	111	77	34	30.63063063
85	111	77	34	30.63063063
86	111	77	34	30.63063063
87	111	77	34	30.63063063
88	101	70	31	30.69306931
89	65	45	20	30.76923077
90	131	90	41	31.29770992
91	131	90	41	31.29770992
92	134	92	42	31.34328358

93	134	92	42	31.34328358
94	136	93	43	31.61764706
95	136	93	43	31.61764706
96	147	100	47	31.97278912
97	84	57	27	32.14285714
98	146	99	47	32.19178082
99	146	99	47	32.19178082
100	136	91	45	33.08823529
101	114	76	38	33.33333333
102	108	72	36	33.33333333
103	133	88	45	33.83458647
104	134	88	46	34.32835821
105	134	88	46	34.32835821
106	122	80	42	34.42622951
107	110	72	38	34.54545455
108	153	100	53	34.64052288
109	153	100	53	34.64052288
110	118	77	41	34.74576271
111	138	90	48	34.7826087
112	138	90	48	34.7826087
113	94	61	33	35.10638298
114	94	61	33	35.10638298
115	144	93	51	35.41666667
116	152	98	54	35.52631579
117	127	81	46	36.22047244
118	102	65	37	36.2745098
119	154	98	56	36.36363636
120	154	98	56	36.36363636
121	154	98	56	36.36363636
122	154	98	56	36.36363636
123	74	47	27	36.48648649
124	74	47	27	36.48648649
125	74	47	27	36.48648649
126	74	47	27	36.48648649
127	156	99	57	36.53846154
128	156	99	57	36.53846154
129	139	87	52	37.41007194
130	55	34	21	38.18181818
131	122	74	48	39.3442623
132	122	74	48	39.3442623
133	139	84	55	39.56834532
134	139	84	55	39.56834532
135	139	84	55	39.56834532
136	139	84	55	39.56834532
137	139	84	55	39.56834532
138	131	79	52	39.69465649
139	131	79	52	39.69465649
140	150	90	60	40
141	154	92	62	40.25974026
142	154	92	62	40.25974026
143	154	92	62	40.25974026
144	154	92	62	40.25974026
145	152	90	62	40.78947368
146	114	67	47	41.22807018
147	143	84	59	41.25874126
148	143	84	59	41.25874126
149	143	84	59	41.25874126
150	143	84	59	41.25874126
151	162	95	67	41.35802469
152	156	91	65	41.66666667
153	156	91	65	41.66666667
154	155	90	65	41.93548387
155	160	92	68	42.5
156	160	92	68	42.5
157	71	40	31	43.66197183
158	174	98	76	43.67816092
159	174	98	76	43.67816092
160	174	98	76	43.67816092
161	174	98	76	43.67816092
162	73	41	32	43.83561644
163	125	70	55	44
164	140	78	62	44.28571429
165	140	78	62	44.28571429
166	167	93	74	44.31137725
167	167	93	74	44.31137725
168	178	99	79	44.38202247
169	178	99	79	44.38202247
170	132	73	59	44.6969697
171	85	47	38	44.70588235
172	85	47	38	44.70588235
173	85	47	38	44.70588235
174	85	47	38	44.70588235
175	125	69	56	44.8
176	182	100	82	45.05494505
177	141	77	64	45.39007092
178	141	77	64	45.39007092
179	141	77	64	45.39007092
180	141	77	64	45.39007092
181	171	93	78	45.61403509
182	72	39	33	45.83333333
183	72	39	33	45.83333333
184	165	89	76	46.06060606
185	165	89	76	46.06060606
186	173	93	80	46.24277457
187	151	81	70	46.35761589
188	102	54	48	47.05882353
189	172	91	81	47.09302326
190	172	91	81	47.09302326
191	123	65	58	47.15447154
192	172	90	82	47.6744186
193	140	73	67	47.85714286
194	75	39	36	48
195	75	39	36	48
196	149	77	72	48.32214765
197	105	54	51	48.57142857
198	159	81	78	49.05660377
199	89	45	44	49.43820225
200	93	47	46	49.46236559
201	93	47	46	49.46236559
202	186	93	92	49.46236559
203	93	47	46	49.46236559
204	93	47	46	49.46236559
205	107	54	53	49.53271028
206	109	55	54	49.5412844
207	129	65	64	49.6124031
208	137	69	68	49.6350365
209	139	70	69	49.64028777
210	149	75	74	49.66442953
211	157	79	78	49.68152866
212	157	79	78	49.68152866
213	161	81	80	49.68944099
214	173	87	86	49.71098266
215	175	88	87	49.71428571
216	175	88	87	49.71428571
217	177	89	88	49.71751412
218	177	89	88	49.71751412
219	179	90	89	49.72067039
220	183	92	91	49.72677596
221	183	93	91	49.72677596
222	183	92	91	49.72677596
223	185	93	92	49.72972973
224	191	96	95	49.7382199
225	191	96	95	49.7382199
226	195	98	97	49.74358974
227	195	98	97	49.74358974
228	195	98	97	49.74358974
229	195	98	97	49.74358974
230	195	98	97	49.74358974
231	199	100	99	49.74874372
232	199	100	99	49.74874372
233	199	100	99	49.74874372
234	199	100	99	49.74874372

235	199	100	99	49.74874372
236	168	84	84	50
237	168	84	84	50
238	134	67	67	50
239	94	47	47	50
240	168	84	84	50
241	148	74	74	50
242	38	19	19	50
243	134	67	67	50
244	168	84	84	50
245	168	84	84	50
246	94	47	47	50
247	154	77	77	50
248	154	77	77	50
249	94	47	47	50
250	168	84	84	50
251	154	77	77	50
252	190	95	95	50
253	154	77	77	50
254	182	91	91	50
255	148	74	74	50
256	38	19	19	50
257	134	67	67	50
258	180	90	90	50
259	184	92	92	50
260	180	90	90	50
261	78	39	39	50
262	152	76	76	50
263	196	98	98	50
264	198	99	99	50
265	190	95	95	50
266	186	93	93	50
267	156	78	78	50
268	162	81	81	50
269	122	61	61	50
270	94	47	47	50
271	168	84	84	50
272	154	77	77	50
273	190	95	95	50
274	168	84	84	50
275	154	77	77	50
276	190	95	95	50
277	182	91	91	50
278	38	19	19	50
279	180	90	90	50
280	144	72	72	50
281	176	88	88	50
282	184	92	92	50
283	146	73	73	50
284	78	39	39	50
285	180	90	90	50
286	184	92	92	50
287	198	99	99	50
288	160	80	80	50
289	156	78	78	50
290	130	65	65	50
291	122	61	61	50
292	68	34	34	50
293	114	57	57	50
294	80	40	40	50
295	182	91	91	50
296	82	41	41	50
297	143	84	84	58.74125874
298	130	77	77	59.23076923
299	148	92	92	62.16216216
300	138	92	91	65.94202899

Appendix B

This appendix contains a table (Table 2) of the results of four hundred random jobs have been used. The optimization ratio of the proposed approach comparing with traditional R.R has been calculated.

Table 2. Quantitative comparison between traditional round robin and the proposed method

X Values	The Proposed Method	Traditional R. R
1	14	14
2	84	168
3	11	11
4	58	58
5	91	183
6	73	73
7	62	154
8	65	65
9	55	139
10	84	168
11	59	143
12	61	61
13	32	32
14	18	65
15	46	93
16	2	49
17	54	54
18	67	134
19	63	63
20	32	32
21	27	74
22	38	85
23	16	16
24	47	94
25	14	14
26	84	168
27	11	11
28	58	58
29	9	98
30	88	177
31	76	165
32	4	4
33	68	137
34	57	57
35	18	18
36	56	125
37	74	149
38	16	16
39	15	15
40	11	11
41	48	122
42	13	87
43	74	148
44	24	98
45	18	18
46	13	13
47	12	12
48	19	38
49	54	54
50	67	134
51	63	63
52	32	32
53	14	14
54	84	168
55	11	11
56	58	58
57	91	183
58	73	73
59	62	154
60	65	65
61	55	139
62	84	168
63	84	143
64	61	61

65	27	74
66	38	85
67	16	16
68	47	94
69	91	138
70	73	73
71	62	154
72	65	65
73	76	174
74	57	57
75	56	154
76	97	195
77	41	118
78	1	1
79	77	154
80	77	130
81	77	154
82	64	141
83	34	111
84	41	41
85	99	199
86	48	48
87	35	135
88	59	59
89	27	74
90	38	85
91	16	16
92	47	94
93	91	183
94	73	73
95	62	154
96	65	65
97	55	139
98	84	168
99	59	143
100	61	61
101	32	32
102	18	65
103	46	93
104	2	49
105	76	174
106	57	57
107	56	154
108	97	195
109	77	154
110	64	141
111	34	111
112	41	41
113	6	101
114	19	114
115	95	190
116	41	136
117	77	154
118	72	149
119	39	39
120	28	28
121	91	182
122	65	156
123	26	117
124	81	172
125	21	21
126	78	159
127	80	161
128	76	76
129	48	122
130	13	87
131	74	148
132	24	98
133	18	18
134	13	13
135	12	12
136	19	38
137	67	134
138	45	45
139	47	114
140	50	50
141	41	131
142	48	138
143	90	180
144	63	63
145	17	109
146	42	134
147	16	16
148	92	184
149	74	167
150	43	136
151	92	186
152	55	55
153	90	180
154	17	107
155	65	65
156	88	88
157	38	138
158	99	199
159	25	125
160	38	138
161	1	1
162	36	75
163	39	78
164	33	72
165	7	62
166	24	79
167	54	109
168	52	52
169	66	66
170	78	157
171	52	131
172	34	113
173	29	105
174	76	152
175	38	114
176	46	46
177	54	152
178	98	196
179	1	99
180	6	6
181	26	118
182	68	160
183	92	148
184	10	102
185	32	132
186	99	199
187	53	153
188	43	143
189	79	178
190	99	198
191	47	146
192	57	156
193	69	139
194	68	68
195	55	125
196	31	101
197	58	123
198	6	71
199	37	102
200	64	129
201	67	162
202	32	127
203	95	190
204	4	99
205	93	186
206	80	173

207	78	171
208	51	144
209	22	100
210	62	140
211	78	156
212	63	63
213	46	127
214	70	151
215	81	162
216	10	91
217	5	93
218	14	102
219	87	175
220	46	134
221	82	82
222	18	116
223	28	28
224	97	195
225	33	94
226	30	30
227	1	62
228	61	122
229	27	74
230	38	85
231	16	16
232	47	94
233	55	139
234	84	168
235	59	143
236	61	61
237	32	32
238	18	65
239	46	93
240	2	49
241	76	174
242	57	57
243	56	154
244	97	195
245	77	154
246	64	141
247	34	111
248	41	41
249	40	136
250	22	118
251	95	191
252	32	128
253	6	101
254	19	114
255	95	190
256	41	136
257	55	139
258	84	168
259	59	143
260	61	61
261	32	32
262	18	65
263	46	93
264	2	49
265	76	174
266	57	57
267	56	154
268	97	195
269	9	98
270	88	177
271	76	165
272	4	4
273	77	154
274	64	141
275	34	111
276	41	41
277	40	136
278	22	118
279	95	191
280	32	128
281	6	101
282	19	114
283	95	190
284	41	136
285	91	182
286	65	156
287	26	117
288	81	172
289	2	20
290	4	17
291	4	16
292	19	38
293	41	131
294	48	138
295	90	180
296	63	63
297	38	110
298	72	144
299	10	82
300	36	108
301	31	119
302	88	176
303	45	133
304	22	22
305	17	109
306	42	134
307	16	16
308	92	184
309	74	167
310	43	136
311	92	185
312	55	55
313	67	140
314	73	146
315	26	99
316	59	132
317	20	20
318	86	173
319	23	110
320	52	139
321	20	65
322	44	89
323	45	45
324	36	36
325	1	1
326	36	75
327	39	78
328	33	72
329	60	150
330	90	180
331	37	37
332	65	155
333	89	179
334	64	64
335	82	172
336	62	152
337	66	66
338	78	157
339	52	131
340	34	113
341	26	118
342	68	160
343	92	184
344	10	102
345	32	132
346	99	199
347	53	153
348	43	143

349	79	178
350	99	198
351	47	146
352	57	156
353	42	122
354	80	160
355	12	92
356	33	33
357	22	100
358	62	140
359	78	156
360	63	63
361	19	84
362	8	73
363	65	130
364	20	85
365	5	93
366	14	102
367	87	175
368	46	134
369	33	94
370	30	30
371	1	62
372	61	122
373	21	55
374	34	68
375	15	49
376	7	7
377	11	68
378	13	70
379	27	84
380	57	114
381	82	182
382	35	135
383	47	147
384	99	199
385	31	71
386	19	19
387	40	80
388	15	55
389	51	105
390	48	102
391	53	107
392	44	44
393	91	182
394	45	136
395	23	114
396	14	105
397	36	36
398	32	73
399	41	82
400	16	16