

An improved round robin time sharing algorithm for optimizing data mapping in cloud computing environments

Afaf Abdelkader¹, Asmaa Mohamed¹, Nermeen Ghazy²

¹Department of Mathematics, Faculty of Science (Girls), Al-Azhar University, Cairo, Egypt

²Department of Information System, Al-Alson Higher Institute, Cairo, Egypt

Article Info

Article history:

Received Dec 25, 2023

Revised Aug 16, 2025

Accepted Sep 1, 2025

Keywords:

Cloud computing

Novel round robin task

scheduling algorithm

Round robin

Scheduling

Time quantum

Time-sharing algorithm

ABSTRACT

Cloud computing in recent years has been widely applied in a wide number of applications and fields. However, allocating tasks to virtual machines (VMs) remains a part that needs enhancement. Task scheduling algorithms in heterogeneous computing system are required to satisfy high-performance data mapping requirements. The efficient allocation between resources and tasks decreases waiting time (WT), turnaround time (TT) and maximizes resource utilization. Various task scheduling algorithms, including round robin (RR) and some improved RR algorithm are used for cloud environment. A novel time-sharing algorithm (NRRTSA) is introduced, demonstrating enhancements in WT and TT. Simulation findings indicate that the NRRTSA algorithm effectively schedules multiple requests (cloudlets) among several VMs, the proposed NRRTSA outperforms RR and other algorithms in terms of the average of both TT and WT. The average turnaround time (ATT) is enhanced with a ratio of 10.8% to 45%, the average waiting time (AWT) is enhanced with a ratio of 10.9% to 45%.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Nermeen Ghazy

Department of Information System, Al-Alson Higher Institute

Cairo, Egypt

Email: nermeen.a2025@gmail.com

1. INTRODUCTION

A cloud constitutes a technological framework that connects several users to multiple systems over a network. Cloud computing employs hardware and software to deliver services via the internet. In cloud computing, users can gain access to files and utilize apps from any internet-enabled device. In a cloud environment, users can store and manage their data easily [1]. The components of cloud computing are available to clients of the cloud on a demand basis. Consumers consistently seek to utilize services that are cost-effective and uninterrupted. The three major types of components available in cloud computing are generally platform as a service (PaaS), software as a service (SaaS), and infrastructure as a service (IaaS) [2]. To optimize a system is to modify it so that it does more of its features, produces results faster, or use less number of resources. Samples of optimization are such as minimizing of waiting time (WT) for clients in a bank before they're addressed by the bank worker; here, the resources are bank worker, the windows, the clients, and the computers. Given that cloud users can access files and applications through the internet from any platform, a primary function of the cloud is resource allocation for processing (i.e., process scheduling), which involves assigning available resources to the necessary cloud applications online to optimize various required parameters [3].

A primary concern in cloud computing is task scheduling. It signifies employing an efficient algorithm to allocate client tasks to suitable and available resources. Scheduling i tasks on j resources is

classified as an NP-hard issue with a runtime complexity of $O(ji)$. Utilizing an efficient task scheduling method is crucial for optimizing system performance. In a cloud computing system, one of the various task scheduling strategies is classical algorithms, such as round robin (RR) [4]. The RR algorithm is a scheduling preemptive technique. It functions effectively in time-sharing. These environments must guarantee acceptable response times (RT) for interactive users. Time slice, or time quantum (TQ), is given to each process in the ready queue (RQ) by the RR scheduling algorithm. Upon the expiration of the TQ, the existing process is moved to the conclusion of this RQ. RR reduces both the average waiting time (AWT) and average turnaround time (ATT). The TQ length is the main problem with the RR approach. Setting a TQ too short result an excessive number of context shifts, which reduces central processing unit (CPU) efficiency. If the TQ is excessively prolonged, the process tends to resemble the first come first served (FCFS) algorithm, potentially resulting in increased reaction time [4]. It is essential to optimize the RR algorithm to decrease AWT and ATT.

This paper's primary contribution is the enhancement of the RR method through the use of TQ in cloud computing to address task scheduling issues. A unique algorithm for progressively altering TQ at different phases of the RQ is proposed. A model of mathematics is constructed to illustrate that the suggested method surpasses the conventional RR technique across various performance parameters, including AWT, and ATT. The results of the experiment show that the more optimized edition of the RR algorithm surpasses the conventional RR algorithm. This suggested technique addresses the issue by employing a progressive TQ, which is derived from the median task burst time in a dynamic fashion. Moreover, the tasks are organized in increasing order, and the proposed algorithm is subsequently implemented for each operation to enhance WT and turnaround time (TT). Relative to the other RR algorithms examined in this research, the drawbacks of the algorithms such as improved round robin (IRR) [5], round robin remaining time (RRRT), and ameliorated round robin algorithm (ARRA) are their propensity to yield elevated AWT and ATT. The objective of this effort is to minimize AWT and reduce ATT. This study consists of five primary sections, with section 2 dedicated to a literature review. Section 3 elucidates the proposed algorithm. The procedure outlined in section 4. Section 5 examines the collected results and discussion, while section 6 addresses the conclusion and future scope.

2. RELATED WORK

Cloud computing presents a vast distributed environment; many algorithms for cloud computing are outlined above. Based Balharith and Alhaidari [4], the RR algorithm distributes CPU time to all processes sequentially, each receiving an equal time interval referred to as TQ. A procedure is obstructed and relegated to the end of the RQ upon the conclusion of a TQ. This procedure is applied to all subsequent processes until their execution is complete, at which point any completed process will be removed from the RQ. RR is effective in the cloud computing environment, enhancing the quality of service (QoS) and system performance for clients. The efficacy of RR is entirely contingent upon the specified TQ. If TQ is chosen to be exceedingly large, the RR algorithm will effectively resemble FCFS. If TQ is excessively small, it will generate significant overhead, resulting in increased average waiting and TTs. Consequently, advancements in technology have led to the establishment of numerous RR algorithms grounded in static or dynamic TQ. The dynamic TQ may be altered either following a cycle or upon the arrival of a new process in the RQ.

Mishra and Khan [5] presented a method known as the IRR. It resembles RR with a minor enhancement, as it chooses the first process that goes into the RQ and assigns it to the CPU for a duration of up to one TQ. Upon the TQ's conclusion, the executing process's remaining duration is assessed; if it is less than one TQ, the CPU is reassigned to the currently executing process for the remainder of the duration. If not, the process will be relegated to the end of the RQ, and the subsequent process will be chosen. It performs more effectively than RR by decreasing both WT and TT. Hiranwal [6] suggested a priority scheduling approach predicated on the burst time of processes. It consists of two phases. The processes are organized based on burst time, prioritizing the process with the shortest burst time. This method selects the optimal time slicing according to the quantity of processes. When the number of processes is even, the optimal time slice will equal the mean of the burst times of all processes. If the number of processes is odd, the optimal time slice corresponds to the burst duration of the mid- process.

According to Sharma and Kakhani [7], the RRRT is presented. It posited that all processes arrive concurrently in the RQ and are subsequently organized in decreasing order according to their burst time. TQ has been calculated using the formula $\sum p_i / 2n$. If the remainder time of the ongoing process is less than the TQ, the CPU is reallocated to the presently executing process. Otherwise, the process will be relegated to the ending of the RQ. The algorithm priority based dynamic round robin (PBDRR) is introduced, which computes an adaptive time slicing for each process and adjusts it subsequent each execution round. The results of experiments showed that the proposed approach surpassed alternative algorithms in decreasing context switches, as well as in reducing both AWT and ATT in [8]. Joshi and Goswami [9] have created an

optimal algorithm termed the “Modulo Based Round Robin Algorithm”. Their algorithm computed a sophisticated TQ. Subsequently, it allocated priorities to the processes and, based on these priorities, calculated the context switches, WT, and TT. The experimental results indicate that, regarding the number of context shifts, as well as the AWT and ATT, the suggested approach outperforms the simple RR approach. The approach developed in [10] rectified all the deficiencies of the RR CPU scheduling mechanism. Furthermore, it provides an analysis that compares the proposed method with the existing RR scheduling algorithm, emphasizing the averages of both WT and TT.

EIDahshan *et al.* [11] presented an algorithm that enhanced the efficacy of certain RR algorithms and devised a TQ that attained stability for the averages of both WT and TT. Stephen *et al.* [12], introduced an improved RR algorithm (RAST ERR approach) to reduce waiting and TTs by utilizing average burst time. The suggested approach surpasses the current RR and alternative algorithms. The proposed approach in [13] simultaneously reduces WT, TT, context shifts, and RT. The calculation of TQ use (1):

$$\text{Time quantum} = \text{mean of processes} + \text{median of processes} \quad (1)$$

where mean and median are the average and the median of execution times of processes respectively. This algorithm reduced the WT, TT, and the number of context switches which in turn improved the system overall performance.

The algorithm referenced in [14] is a dynamic CPU scheduling algorithm. The determination of an appropriate TQ relies on the numeric outlier detection technique and geometric mean when the burst times of processes entering the RQ contain outliers. The experimental results of their method were compared with enhanced variants of RR, demonstrating that the suggested algorithm outperformed the others in terms of AWT and ATT. Dash *et al.* [15] devised an optimized RR CPU scheduling algorithm with a dynamic TQ. It employs a dynamic TQ rather than the static TQ utilized in RR scheduling. The efficacy of the suggested strategy is empirically evaluated against traditional RR and several contemporary alternatives to RR. The findings of our methodology demonstrated enhanced performance in AWT, ATT, and context switching.

Shafi *et al.* [16] introduced an innovative CPU scheduling method called amended dynamic round robin (ADRR), based on CPU burst time. This approach intends to enhance the traditional RR scheduling algorithm with the principle of an active TQ. It periodically modified the TQ in accordance with CPU burst duration. They evaluated the efficacy of their proposed algorithm based on characteristics such as WT and return time. Their simulation findings and numerical analysis in MATLAB indicate that ADRR surpasses other established techniques. Chen *et al.* [17] suggested resource oriented scheduling algorithm (ROSA), which amalgamates both proactive and reactive tactics. Simulation studies were conducted to compare ROSA with five standard algorithms, and the findings indicate that ROSA outperforms the five algorithms in terms of costs, deviation, resource consumption, and fairness.

Alsheikhy *et al.* [18] provide an enhanced dynamic RR scheduling technique utilizing a modified quantum time. A novel enhanced dynamic RR scheduling technique was introduced to reduce AWT, ATT, and the frequency of context shifts, thereby improving overall system performance. It also conducted a comparative analysis of various existing RR algorithms based on AWT, ATT, and the frequency of context switches.

Souza *et al.* [19] suggested an optimal RR CPU scheduling algorithm utilizing Euclidean distance. This offers a novel technique for determining the TQ autonomously by uncovering the correlation between the burst times of all processes in the RQ, utilizing a metric known as Euclidean distance. The similarity metric is employed to identify patterns in the burst times of processes present in the RQ. Arunarani *et al.* [20] examined unique scheduling methodologies to identify which qualities should be incorporated into a certain system and which should be excluded. The literature review is structured according to three distinct perspectives: methodologies, applications, and parameter-based metrics employed. Furthermore, prospective research topics concerning cloud computing-based scheduling are delineated.

According to Rasheed *et al.* [21], cloud-fog infrastructure has transformed the contemporary world by offering low latency, great efficiency, enhanced security, expedited decision-making, and reduced operational costs. The integration of smart grids (SGs) with a cloud-fog platform provides a high-quality source and secure generation, communication, and distribution of power, together with continuous control of demand-supply sequences. This study proposes the use of SG practices into a cloud-fog environment to enhance resource allocation. Six fogs are meticulously analyzed throughout various geographical locations. Each fog is associated with clusters, and each cluster comprises 500 smart homes. To meet the energy demands of residences, fogs accommodate multiple requests, employing various load balancing strategies on virtual machines (VMs) to ensure optimal RT and processing time (PT). The authors propose the max-min algorithm for load balancing with a progressive service broker strategy. Upon evaluating the proposed load balancing algorithm against RR through simulations, we determine that the intended load balancing methods

surpass RR in performance. Chen *et al.* [22] suggested a job scheduler utilizing a dynamic grouping integrated neighboring search technique, which optimizes resource consumption and enhances performance and data locality in heterogeneous computing settings.

Ghazy *et al.* [23] developed a modified multi-level round robin (MMRR) algorithm to substantially improve the efficacy of the standard RR algorithm. The suggested technique determines an appropriate TQ and generates it for each cluster based on the remainder burst time of the processes. The performance has improved regarding latency, PT, and context switching. The results of experiments indicate that the suggested method surpasses alternative algorithms. Jaber *et al.* [24] suggested an optimization model utilizing a multi-objective improved cuckoo search algorithm (MOICS) to enhance task scheduling in a cloud setting by automating the assignment of work to nodes in a cloud. This approach diminished the processing duration for the tasks and the total expenditure. Computational resources for professional use on nodes in a cloud are allocated according to the proposed technique. The proposed technique reduces both cost and makespan.

A hybrid max-min genetic algorithm (HMMGA) is presented in [25] for job scheduling and load balancing. Each VM undergoes an initial load assessment; if the load is elevated, HMMGA is employed for load balancing. The jobs are transferred from the overloaded VMs to the underloaded VMs utilizing the proposed technique HMMGA. In the cloud context, HMMGA significantly mitigates the action imbalance resulting from workload disparity. Ali *et al.* [26] examine studies on CPU scheduling processes to determine the most effective algorithm. Following our examination of the RR, shortest job first, FCFS, and priority algorithms, we discovered that numerous researchers have proposed diverse methodologies for enhancing CPU optimization metrics, such as WT, RT, and TT; however, no single algorithm excels across all criteria. Sharma and Sharma [27] have presented a native task scheduling strategy based on the optimization of the RR scheme. The new approach efficiently enhances CPU utilization by actively establishing an optimal quantum time interval. Experimental results have validated the efficacy of the proposed method compared to conventional RR and its current optimized versions.

Krishnadoss *et al.* [28] introduced the CCSA, an efficient hybrid scheduling method designed to enhance the job scheduling process. It emulates the parasitic behavior of the cuckoo and the food-gathering habits of the crow. The crow consistently monitors its surroundings to identify a superior food source than its current one.

In certain cases, the crow even goes so far as to seize its neighbor's food. The CCSA was developed for use in cloud environments to choose a suitable VM for executing the job scheduling process, inspired by the characteristics of these birds. The planned CCSA decreased both the makespan and cost. Khatri [29], the authors endeavor to devise a novel method that enhances the traditional RR algorithm. The newly suggested method, enhanced round robin (ERR), is compared with the conventional RR algorithm and the improved RR algorithm, with research findings indicating it yields a minimal AWT and ATT.

Abdelkader *et al.* [30] presented a modified median mean round robin (MMMRR) method to improve the efficacy of the RR algorithm. The proposed algorithm determines an optimal dynamic TQ $\lfloor (\text{median} + \text{mean})/2 \rfloor$. The system act has been improved regarding WT, TT, and context switching. The results of experiments demonstrate that the proposed method outperforms alternative methods. An algorithm is developed in [31] that is a highly efficient scheduling mechanism consistent with the relevant computer paradigms. A novel job scheduling mechanism for cloud computing, termed the ARRA, is introduced. The developed approach optimizes the TQ by averaging task burst times through both static and dynamic methods. The investigational results indicated that the ARRA substantially surpassed alternative algorithms, including improved RR, enhanced RR, dynamic TQ approach (ARR), and enhanced RR (RAST ERR), in terms of AWT, ATT, and reaction time. The drawback of numerous algorithms, including RR, IRR, RRRT, and ARRA, is that they result in elevated AWT and ATT. The suggested approach is designed to compute the TQ, assigning it the value of the median, which represents the central execution time. Its objective is to diminish: 1 the mean turnaround duration and the mean waiting duration. This paper primarily proposes a novel method, termed the novel RR time sharing method in cloud computing novel round robin task scheduling algorithm (NRRTSA), which aims to mitigate the limitations of the RR algorithm by enhancing performance metrics through the reduction of AWT and ATT for certain algorithms. This is accomplished by selecting an ideal TQ that minimizes WT and TT.

Zohora *et al.* [32] introduced an optimized NRRTSA adapted for cloud computing environments. The implemented algorithm intends to improve efficiency by dynamically adjusting the TQ for each task execution cycle. The NRRTSA alleviates the disadvantages linked to static TQ allocation, such as heightened context switching and augmented ATT and AWT in cloud computing. The algorithm fundamentally depends on its dynamic TQ computation, which considers the discrepancies among the three maximum burst times of tasks in the RQ for each iteration. The assessment of NRRTSA was performed using three principal metrics: ATT, AWT, and number of context switches (NCS). The results of experiments indicated that NRRTSA significantly reduced ATT, AWT, and NCS across all assessed datasets compared to other recognized RR

methodologies. The study shows that NRRTSA has enhanced performance, acceptability, and optimality in cloud environments, owing to its ability to dynamically alter TQ, leading to improved resource usage and reduced WT and TT for tasks.

Biswas *et al.* [33] presented an enhanced RR with dynamic time quantum (ERRDTQ) method for task scheduling in cloud computing environments, specifically aimed at managing real-time processes characterized by asymmetric burst lengths. Task scheduling is essential in cloud computing for managing the dynamic execution of user requests and resource allocation, significantly contributing to the development of an optimal cloud environment. Traditional RR algorithms frequently encounter difficulties in determining an optimal TQ, resulting in elevated context switching (CS) and diminished CPU efficiency. The main goal of ERRDTQ is to reduce AWT, minimize the number of context switches (CS), decrease ATT, and achieve a balanced distribution of context switches. Experimental results indicate that ERRDTQ surpasses current enhanced RR task scheduling methods in cloud computing environments, achieving a 15.77% reduction in AWTs and a 20.68% decrease in context switching relative to five other improved RR approaches. The research highlights its cost-effectiveness, efficiency, and feasibility in task scheduling and resource allocation within cloud environments.

Al-Shammare and Al-Otaiby [34] implemented a RR algorithm with a smart time quantum (RR-STQ) in a cloud computing environment, utilizing the CloudSim program. The RR-STQ showed a significant increase in average RT, superior performance in TT, WT, and RT compared to the conventional RR algorithm. The dynamic time quantum (DTQ) approach demonstrated superior performance compared to a static TQ. The results suggest that integrating the RR algorithm with other scheduling models, such as shortest job first (SJF), improves WT and TT, and that the DTQ enhances the efficacy of the RR method.

3. THE PROPOSED ALGORITHM

3.1. Problem definition

The RR scheduling algorithm is widely utilized in cloud computing as a pre-emptive scheduling method designed for time-sharing operating systems. The users submit their requests (cloudlets) with the assistance of the cloud management. The cloud manager aggregates service requests from clients, monitors the amount of available VMs in the cloud, and oversees the scheduling policies implemented by the cloud. VMs exhibit heterogeneity, characterized by varying circumstances and computational capacities. The objective is to arrange n cloudlets on a VM to minimize the AWT and ATT.

The proposed NRRTSA goes as follows: the users transmit their requests (cloudlets) and the scheduler assigns these requests to the RQ. The requests are arranged increasingly according to their burst times. The TQ is set to be the median of the burst times of the requests found in the RQ. For calculating the median execution time, the number of processes is tested:

- If there is an odd number of a process, the median is the middle execution time where the rank of median is (2):

$$rank_{median} = \frac{n+1}{2} \quad (2)$$

- If there is an even number of processes, the median will be the average of the two central execution times. The ranks of these numbers are $\frac{n}{2}$ and $\frac{n}{2}+1$. TQ is then calculated using (3):

$$TQ = median \quad (3)$$

The requests are evaluated sequentially in ascending order of burst times. If its burst time is less than or equal to TQ, it is processed in full on a VM and thereafter removed from the RQ. Alternatively, perform the request on a VM for a duration equivalent to TQ, and thereafter assess the remainder time of the presently executing process. If the remainder time of the presently executing process is less than or equal to time 1 TQ, it is executed to completion and thereafter removed from the RQ. Otherwise, the leftover portion is placed at the ending of the RQ. This procedure is reiterated until all requests are fulfilled. Upon completion of all requests, the AWT and ATT are computed. The flowchart of the NRRTSA algorithm is illustrated in Figure 1.

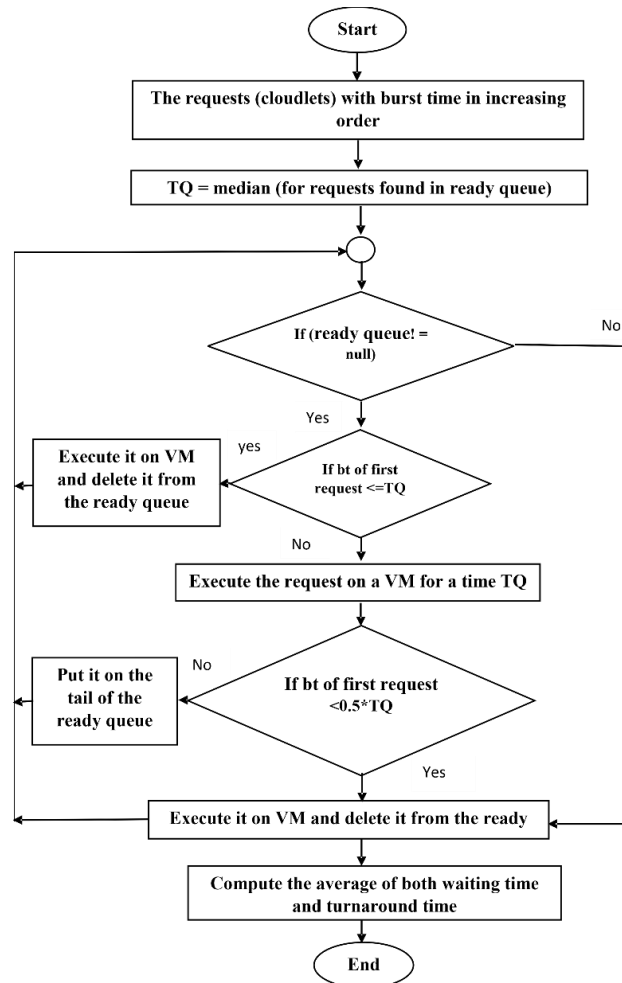


Figure 1. Flowchart of the NRRTSA algorithm

3.2. Pseudo code of NRRTSA

The pseudo code for the NRRSTA is given below.

```

1. Ready queue ← the requests (cloudlets) with burst time in increasing order
2. TQ ← median (median for requests found in ready queue)
3. While (ready queue != null)
4.   Assign 1 TQ to each request in ready queue
5.   If (burst time of request) ≤ TQ → execute it on a VM until completion,
     then delete it from the ready queue
6.   Else
       execute the request on a VM for a time = TQ.
       If (the remaining time of the currently running request < 0.5 TQ) → then
         it is executed until its completion and then it is deleted form
         ready queue.
       Else
         it is put at the tail of ready queue.
7.   Go to step 3
8. End While
9. Compute the average of both waiting time and turnaround time.
  
```

4. METHOD

Users submit several queries simultaneously in cloud computing. Both distinct and identical arrival times can be accommodated by the RR job scheduling method. A variety of tasks are applied to the RR [4], improved RR [5], RR remaining time [7], ARRA [31], and NRRTSA algorithms in order to assess the efficiency of the suggested approach. Here we give a comparative analysis of various algorithms. The main criteria for comparison are AWT and ATT.

The AWT is calculated as (4):

$$AWT = \Sigma (TAT \text{ of } P_i - BT \text{ of } P_i) / N \quad (4)$$

The ATT is calculated as (5):

$$ATT = \Sigma (T_i - AT \text{ of } P_i) / N \quad (5)$$

Where ATT of P_i is the TT of task P_i , BT of P_i is the burst time of task P_i , T_i is the exit time for task, AT of P_i is the arrival time of task and N is the number of all tasks. Two scenarios are utilized to test the efficiency of the suggested algorithm.

4.1. Scenario 1 (zero arrival time)

Here, we considered the situation where all tasks are arrived at the same time. Consider a set of 6 requests with burst time shown in Table 1, assigned to the RQ. Figures 2 through 6 illustrate the performance of the proposed algorithms in sequential order. Specifically, Figure 2 presents the RR algorithm, Figure 3 presents the IRR algorithm, Figure 4 presents the RRRT algorithm, Figure 5 presents the ARRA algorithm, and Figure 6 presents the NRRTSA algorithm.

Table 1. Input burst time for requests

Requests	Burst time (ms)
P0	8
P1	3
P2	11
P3	19
P4	2
P5	30

4.1.1. Basic round robin algorithm [4]

Enter these requests in RQ according to their given order of burst time that is $P_0=8$, $P_1=3$, $P_2=11$, $P_3=19$, $P_4=2$, and $P_5=30$ with $TQ=10$. The Gantt chart for it is given as shown in Figure 2.

Gantt chart:

P0	P1	P2	P3	P4	P5	P2	P3	P5	P5	
0	8	11	21	31	33	43	44	53	63	73

Figure 2. Gantt chart for the basic RR algorithm

Resulting $AWT=24.83$ ms and resulting $ATT=37$ ms.

4.1.2. Improved round robin algorithm [5]

Put the requests in the RQ in their order; $P_0=8$, $P_1=3$, $P_2=11$, $P_3=19$, $P_4=2$, and $P_5=30$ with $TQ=10$. Each request is given a TQ to execute. The remaining time of the currently running request is checked; if it is less than 1 TQ, the CPU is again allocated to the currently running request for the residual time. Otherwise, the request will be put at the end of the RQ and the next request will be selected. The Gantt chart for it is given as shown in Figure 3.

Gantt chart:

P0	P1	P2	P2	P3	P3	P4	P5	P5	P5	
0	8	11	21	22	32	41	43	53	63	73

Figure 3. Gantt chart for the algorithm IRR

Resulting $AWT=20.8$ ms and resulting $ATT=33$ ms.

4.1.3. Round robin remaining time algorithm [7]

Put the requests in the RQ in an increasing order of burst time; $p_4=2$, $p_1=3$, $p_0=8$, $p_2=11$, $p_3=19$, and $p_5=30$. The TQ is calculated according to RRRT algorithm from the following formula $\sum p_i/2n$ where its value is 6. The remaining time of the currently running request is checked where if it is <1 TQ, the CPU is again allocated to the currently running request for the residual time. Otherwise, the request will be put at the end of the RQ and the next request will be selected. The Gantt chart for it is given as shown in Figure 4.

Gantt chart:

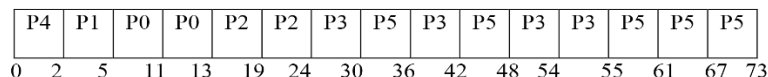


Figure 4. Gantt chart for the algorithm RRRT

Resulting AWT=16.5 ms and resulting ATT=28.6 ms.

4.1.4. The ameliorated round robin algorithm [31]

Put the processes in an increasing order of burst time; $p_4=2$, $p_1=3$, $p_0=8$, $p_2=11$, $p_3=19$ and $p_5=30$. The TQ that is calculated for this algorithm using the formula $TQ = (3/4) * \text{average}$. The Gantt chart for this algorithm is given as shown in Figure 5.

Gantt chart:

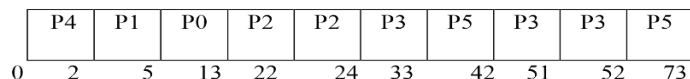


Figure 5. Gantt chart for the algorithm ARRA

Resulting AWT=16 ms and resulting ATT=28.16 ms.

4.1.5. New time-sharing algorithm

Will enter processes RQ according to their increasing order of burst time; $p_4=2$, $p_1=3$, $p_0=8$, $p_2=11$, $p_3=19$, and $p_5=30$. The TQ that is calculated for this algorithm from the formula $TQ = \text{median} = 10$. The Gantt chart for this algorithm is given as shown in Figure 6.

Gantt chart:

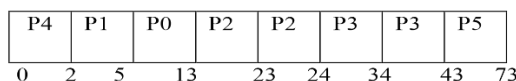


Figure 6. Gantt chart for the algorithm NRRTSA

Resulting AWT=14.5 ms and resulting ATT=26.6 ms.

A comparative study among the RR, IRR, RRRT, ARRA, and NRRTSA algorithms with respect to TQ, AWT, and ATT as represented in Table 2. Figure 7 is comparing the performance of the aforementioned algorithms using AWT and ATT. Figure 7 shows the comparison of AWT where the best result 14.5 ms is obtained from NRRTSA because it gives the minimum value between all other algorithms 24.8, 20.8, 16.5, and 16 ms of basic RR [4], IRR [5], RRRT [7], and ARRA algorithm [31] respectively.

Table 2. Obtained results for algorithms

Algorithm	AWT	ATT
Basic RR	24.83	37
IRR	20.8	33
RRRT	17.5	28.6
ARRA	16	28.16
NRRTSA algorithm	14.5	26.6

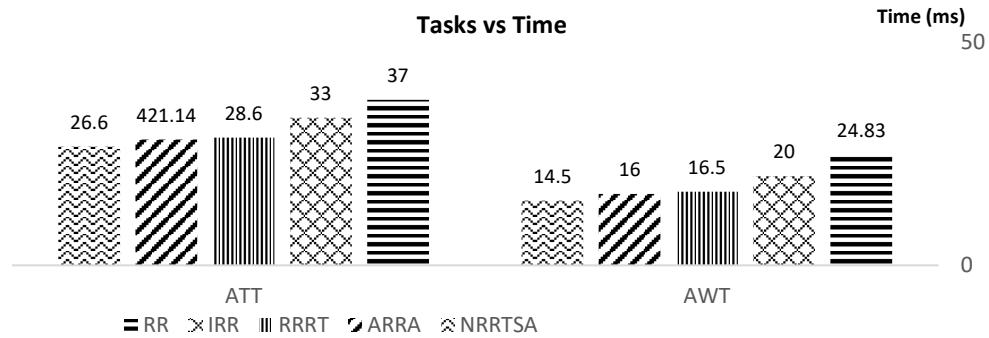


Figure 7. AWT and ATT comparison graph (scenario 1)

Whereas a comparison of the ATT for the five algorithms with best result obtained from the proposed NRRTSA. It obtained a value of 26.6 ms between all other values 37, 33, 28.6, and 28.16 ms of basic RR [4], IRR [5], RRRT [7], and ARRA algorithm [31] respectively.

4.2. Scenario 2 (non-zero arrival time)

In scenario 2, the suggested technique uses a dynamic TQ since the tasks in the RQ have non-zero arrival delays. Every round, TQ is updated. Table 3 illustrates how five requests—P1, P2, P3, P4, and P5—are combined with CPU burst time and non-zero arrival timings. The RR, IRR, RRRT, ARRA, and the suggested NRRTSA algorithms are displayed in Figures 8-12.

Table 3. The burst time and its arrival time for requests

Requests	Arrival time (ms)	Burst time (ms)
P1	0	30
P2	4	20
P3	8	50
P4	12	90
P5	16	65

- Basic RR algorithm [4]

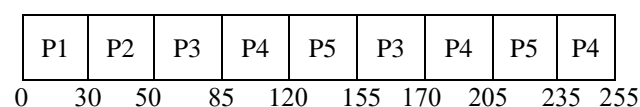


Figure 8. Gantt chart for the algorithm RR

- Improved RR algorithm [5]

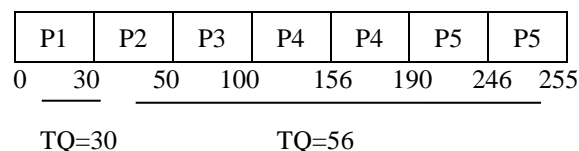


Figure 9. Gantt chart for the algorithm improved RR

- RR remaining time algorithm [7]

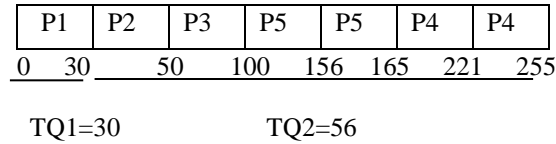


Figure 10. Gantt chart for the algorithm RRRT

- The ARRA [31]

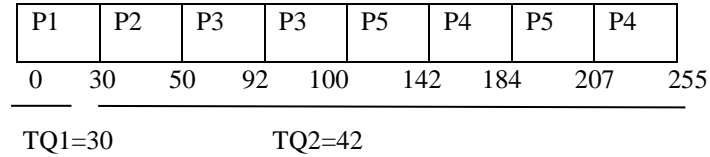


Figure 11. Gantt chart for the algorithm ARRA

- New time-sharing algorithm (NRRTSA)

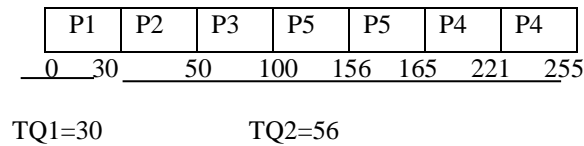


Figure 12. Gantt chart for the algorithm NRRTSA

In Table 4, the RR, IRR, RRRT, ARRA, and NRRTSA algorithms are compared with regard to TQ, AWT, and ATT. Figure 13 shows the comparison of AWT and ATT for the earlier methods in scenario 2.

Table 4. Comparative study of RR, IRR, RRRT, ARRA and NRRTSA algorithms (scenario 2)

Algorithm	TQ	AWT	ATT
RR	35	87	138
IRR	30,56	83	117
RRRT	30,56	61	112
ARRA	30,42	73.6	120.4
NRRTSA	30,56	61	112

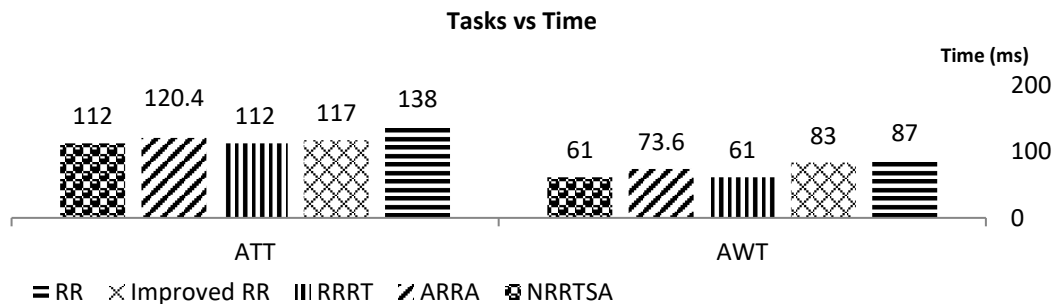


Figure 13. AWT and ATT comparison graph (scenario 2)

4.3. Simulation

The benchmark was reproduced using the same parameters and different algorithms, taking into account the task arrival time and burst time, in order to assess the efficiency of the suggested model. A C++ simulation is created for the proposed algorithm (NRRTSA), the improved RR algorithm [5], the RR remaining time algorithm [7], and the ARRA algorithm [31]. The performance of each of these algorithms is assessed by comparing it to the RR algorithm. With the same random data set, the model consists of a singular resource represented by VMs. These algorithms are compared using AWT and ATT. An escalation in time leads to an increase in cost because the average waiting and TT is dependent on the quantity of tasks in the RQ.

5. RESULTS AND DISCUSSION

To compare the effectiveness of the algorithms, different simulations are made for them using C++ language on laptop with Intel Core i5 processor, using 2.30 GHz CPU and 4 GB RAM. Windows 10 was utilized as the 64-bit OS for the platform.

These models consider n requests, where n ranges from 1000 to 5000. Every task arrives at the same time, and the burst time task values are produced at random. The suggested model was validated through a variety of tests, which are repeated again as the number of tasks increases.

Figure 14 displays the algorithms' comparison in terms of AWT. The stacked line chart is plotted for tasks ranging from 1000 to 5000. The x-axis plots the number of tasks in the RQ, and the y-axis plots the tasks' AWT, which is given in milliseconds. Better results are obtained by the suggested algorithm (NRRTSA), which is followed by ARRA [31], RRRT [7], and IRR [5]. When compared to the RR algorithm, these techniques provide a significant improvement. The performance of the algorithms enhances as the number of tasks in the RQ grows. While the IRR [5] yields respectable improvement outcomes, ARRA [31] and RRRT [7] yield significant outcomes when compared to RR. In contrast, the suggested method outperforms the others in terms of improvement. In comparison to other algorithms, NRRTSA's performance demonstrated an upward trend in AWT as the number of tasks increased. The line chart shows that the AWT for RR is continuously rising in contrast to recommended algorithms. Figure 15 illustrates a similar pattern to the behavior of algorithms in terms of ATT. The stacked line chart is plotted for tasks ranging from 1000 to 5000. The x-axis plots the number of tasks in the RQ, while the y-axis plots the tasks' ATT, which is given in milliseconds. Better results are obtained by the suggested algorithm NRRTSA, which is followed by ARRA [31], RRRT [7], and IRR [5]. When compared to the RR algorithm, these techniques provide a significant improvement. The algorithms' performance improves with the amount of tasks in the RQ. While IRR yields reasonable improvement outcomes, ARRA [31] and RRRT [7] yield considerable results when compared to RR. The suggested algorithms behave similarly, although NRRTSA's performance exhibited an upward trend in ATT when the number of tasks increased in comparison to other algorithms. The line chart shows that the ATT for RR is continuously rising in contrast to recommended algorithms.

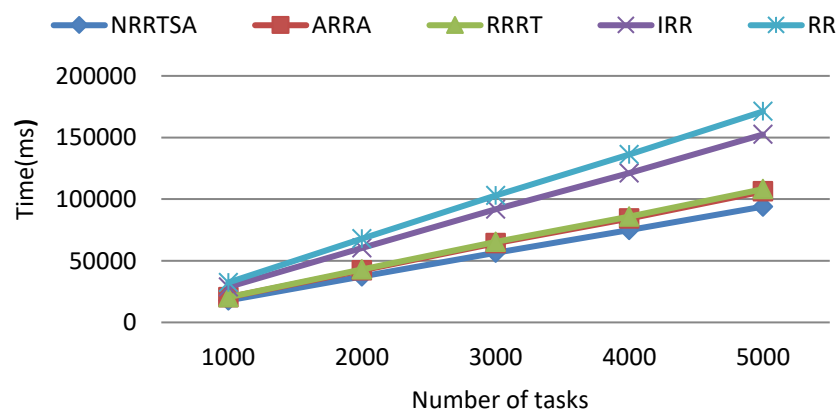


Figure 14. Comparative graph of awaiting time

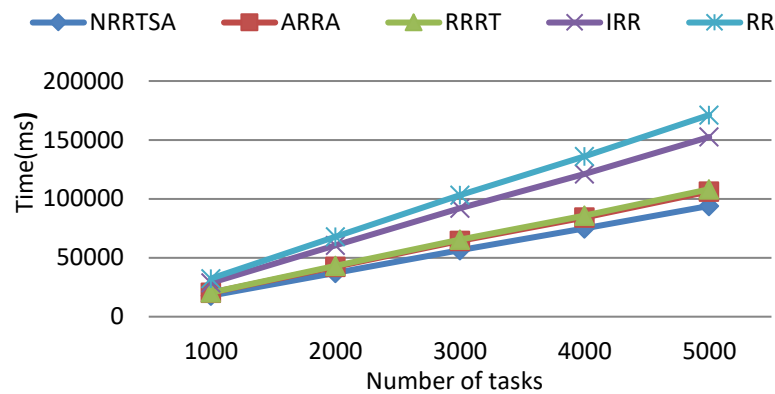


Figure 15. Comparative graph of ATT

6. CONCLUSION

Scheduling requests of users in cloud environment is very important and evaluated here. In this paper, in order to enhance the cloud computing performance metrics of AWT and ATT, the NRRTSA has been suggested. The suggested NRRTSA method demonstrated that the optimal TQ to assign to the tasks in ascending order is (TQ=median) of the processes (requests) identified in the RQ for tasks that arrive simultaneously and at various times. The RR, IRR, RRRT, and ARRA algorithms are used to simulate and compare it. According to the findings of the testing, the suggested NRRTSA performs better than RR and other algorithms in terms of the AWT and TT. A ratio of 10.9% to 45% is used to improve the AWT, and a ratio of 10.8% to 45% is used to improve the ATT. The suggested algorithm may be used to improve other criteria, including context switching, in subsequent research.

ACKNOWLEDGMENTS

The authors express their gratitude to the anonymous reviewers and editors for their insightful comments.

FUNDING INFORMATION

This research received no external funding. The study was self-funded by the authors.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Afaf Abdelkader	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	✓
Asmaa Mohamed		✓				✓		✓	✓	✓	✓			✓
Nermeen Ghazy	✓		✓	✓			✓			✓	✓	✓	✓	✓

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nvestigation

R : **R**esources

D : **D**ata Curation

O : Writing - **O**riginal Draft

E : Writing - Review & **E**ding

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.




REFERENCES

- [1] N. K. Pandey, "Extended Multi Queue Job Scheduling in Cloud," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 15, no. 11, pp. 1–8, 2017.
- [2] P. S. and A. Kumar, "Efficient Resource Utilization in Virtual Cloud Computing Environment," *International Journal of Computer Applications*, vol. 168, no. 11, pp. 25–27, 2017, doi: 10.5120/ijca2017914541.
- [3] K. Eldahshan, A. A. Elkader, and N. Ghazy, "Round Robin based Scheduling Algorithms, A Comparative Study Abstract: 2. Static Time Quantum Algorithms Description of the Round Robin algorithms with assumed Time quantum," *Automatic Control and System Engineering Journal*, vol. 17, no. 2, pp. 29–42, 2017.
- [4] T. Balharith and F. Alhaidari, "Round Robin Scheduling Algorithm in CPU and Cloud Computing: A review," in *2nd International Conference on Computer Applications and Information Security, ICCAIS*, 2019, pp. 1-7, doi: 10.1109/CAIS.2019.8769534.
- [5] M. K. Mishra and A. K. Khan, "An Improved Round Robin CPU Scheduling Algorithm," *Journal of Global Research in Computer Science*, vol. 3, no. 6, pp. 64-69, 2012.
- [6] S. Hiranwal, "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice," *International Journal of Computer Science and Communication*, vol. 2, no. 2, pp. 319–323, 2011.
- [7] A. Sharma and G. Kakhani, "Analysis of Adaptive Round Robin Algorithm and Proposed Round Robin Remaining Time Algorithm," *International Journal of Computer Science and Mobile Computing*, vol. 4, no. 12, pp. 139–147, 2015.
- [8] R. Mohanty, H. S., K. Patwari, M. Dash, and L. Prasanna, "Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 2, 2011, doi: 10.14569/ijacsa.2011.020209.
- [9] A. Joshi and S. Goswami, "Modified Round Robin Algorithm by Using Priority Scheduling," *Advances in Computational Sciences and Technology*, vol. 10, no. 6, pp. 1543–1549, 2017.
- [10] S. Zouaoui, L. Boussaid, and A. Mtibaa, "Priority based round robin (PBRR) CPU scheduling algorithm," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 1, pp. 190–202, 2019, doi: 10.11591/ijece.v9i1.pp190-202.
- [11] K. Eldahshan, A. Abd, and N. Ghazy, "Achieving Stability in the Round Robin Algorithm," *International Journal of Computer Applications*, vol. 172, no. 6, pp. 15–20, 2017, doi: 10.5120/ijca2017915161.
- [12] A. Stephen, B. H. Shanthan, and D. Ravindran, "Enhanced Round Robin Algorithm for Cloud Computing," *International Journal of Scientific Research in Computer Science Applications and Management Studies*, vol. 7, no. 4, pp. 1–5, 2018.
- [13] B. Fataniya and M. Patel, "Dynamic Time Quantum Approach to Improve Round Robin Scheduling Algorithm in Cloud Environment," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 4, no. 4, pp. 963–969, 2018.
- [14] T. O. Omotehinwa, I. Azeez, and E. O. Oyekeani, "An Improved Round Robin CPU Scheduling Algorithm for Asymmetrically Distributed Burst Times," *Africa Journal Management Information System*, vol. 1, no. 4, pp. 50–68, 2019.
- [15] A. 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*, vol. 5, no. 1, pp. 7–26, 2015, doi: 10.5121/ijceit.2015.5102.
- [16] U. Shafi *et al.*, "A novel amended dynamic round robin scheduling algorithm for timeshared systems," *International Arab Journal of Information Technology*, vol. 17, no. 1, pp. 90–98, 2020, doi: 10.34028/iajit/17/1/11.
- [17] H. Chen, X. Zhu, G. Liu, and W. Pedrycz, "Uncertainty-Aware Online Scheduling for Real-Time Workflows in Cloud Service Environment," *IEEE Transactions on Services Computing*, vol. 14, no. 4, pp. 1167–1178, 2021, doi: 10.1109/TSC.2018.2866421..
- [18] A. Alsheikhy, R. Ammar, and R. Elfouly, "An improved dynamic Round Robin scheduling algorithm based on a variant quantum time," in *2015 11th International Computer Engineering Conference: Today Information Society What's Next?, ICENCO, IEEE*, Dec. 2016, pp. 98–104, doi: 10.1109/ICENCO.2015.7416332.
- [19] M. D. Souza, F. Caiero, and S. Surlakar, "Optimal Round Robin CPU Scheduling Algorithm using Euclidean Distance," *International Journal of Computer Applications*, vol. 96, no. 18, pp. 8–11, 2014, doi: 10.5120/16892-6930.
- [20] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computer Systems*, vol. 91, pp. 407–415, 2019, doi: 10.1016/j.future.2018.09.014.
- [21] S. Rasheed, N. Javaid, S. Rehman, K. Hassan, F. Zafar, and M. Naeem, "A Cloud-Fog Based Smart Grid Model Using Max-Min Scheduling Algorithm for Efficient Resource Allocation," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 22, pp. 273–285, 2019, doi: 10.1007/978-3-319-98530-5_23.
- [22] C. T. Chen, L. J. Hung, S. Y. Hsieh, R. Buyya, and A. Y. Zomaya, "Heterogeneous Job Allocation Scheduler for Hadoop MapReduce Using Dynamic Grouping Integrated Neighboring Search," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 193–206, 2020, doi: 10.1109/TCC.2017.2748586.
- [23] N. Ghazy, A. Abdelkader, M. S. Zaki, and K. A. E. Dahshan, "A New Round Robin Algorithm for Task Scheduling in Real-time System," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 5, pp. 691–704, 2022, doi: 10.22266/ijies2022.1031.59.
- [24] S. Jaber, Y. Ali, and N. Ibrahim, "An Automated Task Scheduling Model Using a Multi-objective Improved Cuckoo Optimization Algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 1, pp. 295–304, 2022, doi: 10.22266/IJIES2022.0228.27.
- [25] J. Varghese and J. Sreenivasaiah, "Entropy Based Monotonic Task Scheduling and Dynamic Resource Mapping in Federated Cloud Environment," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 1, pp. 235–250, 2022, doi: 10.22266/IJIES2022.0228.22.
- [26] 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. 19–26, 2021, doi: 10.22937/IJCSNS.2021.21.1.4.




- [27] P. Sharma and Y. M. Sharma, "An Efficient Customized Round Robin Algorithm for CPU Scheduling," in *Lecture Notes in Networks and Systems*, pp. 623–629, 2021, doi: 10.1007/978-981-15-9689-6_68.
- [28] P. Krishnadosh, G. Natesan, J. Ali, M. Nanjappan, P. Krishnamoorthy, and V. K. Poornachary, "CCSA: Hybrid Cuckoo Crow Search Algorithm for Task Scheduling in Cloud Computing," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 241–250, 2021, doi: 10.22266/ijies2021.0831.22.
- [29] J. Khatri, "An Enhanced Round Robin CPU Scheduling Algorithm," *IOSR Journal of Computer Engineering*, vol. 18, no. 04, pp. 20–24, 2016, doi: 10.9790/0661-1804022024.
- [30] A. Abdelkader, N. Ghazy, M. S. Zaki, and K. A. E. Dahshan, "MMRR: a Modified Median Mean Round Robin Algorithm for Task Scheduling," *International Journal of Intelligent Engineering and Systems*, vol. 15, no. 6, pp. 599–608, 2022, doi: 10.22266/ijies2022.1231.53.
- [31] N. Ghazy, A. Abdelkader, M. S. Zaki, and K. A. E. Dahshan, "An ameliorated Round Robin algorithm in the cloud computing for task scheduling," *Bulletin of Electrical Engineering and Informatics*, vol. 12, no. 2, pp. 1103–1114, 2023, doi: 10.11591/eei.v12i2.4524.
- [32] M. F. Zohora, F. Farhin, and M. S. Kaiser, "An enhanced round robin using dynamic time quantum for real-time asymmetric burst length processes in cloud computing environment," *PLoS ONE*, vol. 19, no. 8, pp. 1–18, Aug. 2024, doi: 10.1371/journal.pone.0304517.
- [33] D. Biswas, M. Samsuddoha, M. R. A. Asif, and M. M. Ahmed, "Optimized Round Robin Scheduling Algorithm Using Dynamic Time Quantum Approach in Cloud Computing Environment," *International Journal of Intelligent Systems and Applications*, vol. 15, no. 1, pp. 22–34, 2023, doi: 10.5815/ijisa.2023.01.03.
- [34] H. Al-Shammare and N. Al-Otaiby, "An Implementation of a New Proposed Round-Robin Algorithm with Smart Time Quantum in Cloud Computing Environment," in *Proceedings - 2022 14th IEEE International Conference on Computational Intelligence and Communication Networks, CICN 2022*, 2022, pp. 289–296, doi: 10.1109/CICN56167.2022.10008330.

BIOGRAPHIES OF AUTHORS






Afaf Abdelkader    received the M.Sc. degree in Computer science from Department of Mathematics, Faculty of Science, Al-Azhar University, Egypt, in 2006 and Ph.D. in Computer Science from Department of Mathematics, Faculty of Science, Al-Azhar University, Egypt in 2011. She is currently an associate professor at Al-Azhar University. She can be contacted at email: afaf2azhar@azhar.edu.eg.



Asmaa Mohamed    received the B.Sc. degree in science, in 2011, the M.Sc. degree in computer science, in 2017, and the Ph.D. degree in computer science, in 2023. She is currently an assistant professor in computer science at the Department of Mathematics, Faculty of Science, Al-Azhar University, Cairo, Egypt. She has published several research papers in the field of AI, machine learning, IoT, and data mining. She can be contacted at email: asmaamohamed89@azhar.edu.eg.



Nermeen Ghazy    received the M.Sc. degree in Computer science from Department of Mathematics, Faculty of Science, Al-Azhar University, Egypt, in 2017 and Ph.D. in Computer Science from Department of Mathematics, Faculty of Science, Al-Azhar University, Egypt in 2023. She is currently an assistant professor at Al-Azhar higher institute. She can be contacted at email: nermeen.a2025@gmail.com.