Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Dynamic Adjacent Quantum Time Round Robin (DAQRR) for CPU Scheduling

Ashiru Simon[1], Gabriel Lazarus Dams[2], Salome Danjuma[3]
[1, 2, 3] Department of Computer Science, Kaduna State University, Kaduna, Kaduna State – Nigeria
a.simon@kasu.edu.ng[1], damsgabe@kasu.edu.ng[2], s.danjuma@kasu.edu.ng[3]

Abstract

Round Robin (RR) CPU Scheduling involves allocating CPU resources to processes in a circular manner using a fixed quantum time (QT). Each process in the ready queue receives an equal time slice known as the quantum time, after which the CPU is reassigned to the next process in line. Some processes under RR complete execution while others return to the end of the queue for the next round. However, RR has drawbacks due to its fixed QT: a large QT can mimic the behavior of a First Come First Serve (FCFS) algorithm, while a small QT can lead to excessive context switching.
This research proposes a Dynamic Adjacent Quantum Time Round Robin (DAQRR) scheduling algorithm to dynamically adjust the quantum time without sorting the processes. A simulation was developed using Python programming language to evaluate and compare the performance of DAQRR against classical RR. The proposed DAQRR algorithm demonstrated a 5.7% reduction in average waiting time and a 4.9% reduction in average turnaround time with 10 processes. Furthermore, the algorithm showed improved performance as the number of processes increased to 60, 70, 80, and 90, as detailed in Table 2.

Keywords: *Adjacent Processes; Quantum Time (QT); Burst Time; Dynamic Round Robin (RR); CPU Scheduling.*

1. Introduction

Resources are so important to a system in process scheduling and the central processing unit (CPU) is considered the most significant resource used by the operating system (OS) (Ahmed & Muquit, 2016; Danjuma et al., 2021). The OS is responsible for controlling and coordinating resources to increase the efficacy of applications in a system, otherwise known as scheduling (Mody & Mirkar, 2019; Harki et al., 2020). As processes competes for the CPU, the OS manages the competition using a technique also known as CPU scheduling (Simon et al., 2014a). CPU scheduling involves allocating CPU resources to processes at specific time intervals whereas, the OS responsibly manages all processes that are ready and waiting for execution by the CPU (Shafi et al., 2020; Abubakar et al., 2023).

2. Overview of Some Common CPU Scheduling Algorithms

The aim of CPU scheduling algorithms according to Singh and Agrawal (2017) is to minimize the waiting time (WT), turnaround time (TAT), response time (RT) and number of context switch (CS) which by implication will improve the performance of CPU scheduling. There are numerous proposed CPU scheduling algorithms with some common ones such as First-Come-First-Serve (FCFS), Shortest Job First (SJF), Priority and Round Robin (RR). Each algorithm have its advantage and disadvantage (Ali et al., 2021; Harki et al, 2020; Joshi & Goyal, 2019; Seemakuthi, et al., 2015).

For instance, in FCFS algorithm, processes are assigned CPU according to their arrival time. The first to arrive is assigned the CPU for execution followed by the next process. This implies that the algorithm will always experience

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

higher average waiting time if processes with larger burst time (BT) arrives earlier than those with lesser BT (Farooq et al., 2017). Meanwhile, SJF algorithm allocates CPU based on processes' size unlike the FCFS technique. In SJF, the process with the shortest BT is assigned the CPU first and then followed by the next process with shortest BT (Paul et al., 2019). One disadvantage with this algorithm is starvation. Processes with larger BT will be starved if processes with smaller BTs keeps arriving into the ready queue. The priority algorithms works such that each process is assigned a priority and the CPU is allocated to the process with the highest priority (Seemakuthi et al., 2015). When two or more process have equal priorities, the FCFS algorithm is used for the scheduling. However, the issue of indefinite blocking also known as starvation is the major problem with this algorithm where processes with low priority will be made to indefinitely wait for their turn to use the CPU. Though this problem has been tackled by a technique known as aging – that is, the longer the process wait, the more its priority increases.
Round Robin (RR) is one of the oldest CPU scheduling algorithms, and it has found its importance in time-sharing systems (Joshi & Goyal, 2019; Simon et al., 2014b). In RR algorithm, equal time period known as quantum time (QT) is assigned circularly to each of the processes in the ready queue (Indusree & Prabadevi, 2017). A process is assigned CPU for a particular QT which will run to completion if and only if the QT is less than its BT, otherwise it will relinquish the CPU for the next ready process while it returns to the tail of the queue for the next round. The problem of starvation is solved in this algorithm because every process will get equal time frame for execution. However, selecting the appropriate QT to minimize context switch becomes the big problem in this algorithm being that it is designed especially for time-sharing systems. If QT is too large the system will run just as First Come First Served algorithm while if QT is too small it will create too much context switch. RR has several disadvantages because of its fixed QT. This is why researchers have proposed dynamic QT where the time slot allocated to processes for execution varies dynamically (Al-Bakhrani et al., 2020; Fiad et al., 2020; Shafi et al., 2020; Mora et al., 2017).

3. Process States

A process is a program in execution (Silberschatz et al., 2005). It describes how far the execution of a program has gone and can be in five different states – new, ready, running, terminated and waiting (Zouaoui et al., 2016).

1. **New:** new process is created.
2. **Ready:** the process is ready but waiting to be assign the CPU.
3. **Running:** the process is currently being executed.
4. **Terminated:** The process has finish execution.
5. **Waiting:** process is waiting because of input/output event.

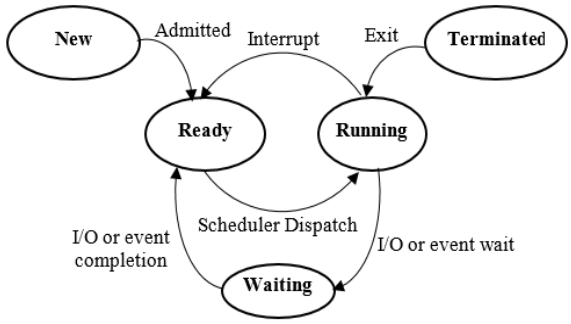


Figure. 1: Process State Diagram
Source: (Silberschatz et al., 2005)

Figure 1 above describes process state as soon as it is created. When a new process is created, it is admitted into the ready queue and wait to be assigned CPU. A process in the ready state may change to running state if the scheduler selects the process from the ready queue using some form of algorithm and allocate it to the CPU for execution. A running process can change its state to terminated, ready or waiting. A process upon completion of its execution will change its state to *terminated*. It can also change its state from running to ready as a result of interrupt generated

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

or from running state to waiting state as a result of input/output request. Upon completion of an input/output operation, a process can change its state from waiting state to ready state.

4. Evaluation Criteria

Many criteria according to Ali et al. (2021) have been suggested for comparing CPU scheduling algorithms. These criteria are used to check the performance of the algorithm. The most common criteria used for evaluation by researchers are waiting time, turnaround time, and number of context switching (Jbara, 2019; Singh & Agrawal, 2017).

1. **Waiting Time**: This is the total time taken by a process waiting in the ready queue to acquire the CPU. These criteria should be minimized.
2. **Turnaround Time**: It is the total time taken by a process before completing its execution from the time it is submitted. It is the sum of time spent waiting to get into memory, have access to the ready queue, executing on the CPU and doing I/O events (Silberschatz et al., 2005). These criteria should be minimized.
3. **Number of context switching:** This is the act of switching the CPU from one process to another. Switching is a pure system overhead cost but still useful in time sharing systems (Silberschatz et al., 2005)

5. Review of Related Works

Many techniques have been proposed by several researchers with an aim of improving the classical round-robin (RR) (Simon et al., 2022; Shafi et al., 2020; Mody & Mirkar, 2019; Srujana et al., 2019; Zouaoui et al., 2019; Biswas et al., 2018).

More so, Majedkan et al., (2020) mentioned that it is necessary to clearly examine and understand related works concerning different scheduling techniques, identify the problems with the reviewed techniques to effectively develop a new technique that will improve the performance of the CPU scheduling. One of the challenges of classical RR is the static nature of its quantum time (QT) – it is fixed throughout program execution. Many researchers have made effort to dynamically assign QT to processes in order to improve the classical RR scheduling by maximizing CPU utilization, throughput and minimizing average waiting time, response time, and turnaround time.

Iqbal et al. (2023) proposed a dynamic RR technique where the time quantum for each process is based on its priority, with higher-priority processes being given a more significant time quantum. It allows the system to prioritize critical processes while ensuring all processes get a fair share of CPU time. 1 is the least priority value. If priority is greater than 1, the algorithm reduces the priority value till it finishes execution – a technique to solve aging.

Shafi et al., (2020) for instance proposed a RR scheduling system in which quantum time (QT) is allocated dynamically to processes in the ready queue and compared it against five different research works. In their works, all processes were assumed to be in the ready queue and were sorted in ascending order. The QT for each round is equal to the burst time of the least process, provided the least process BT is greater than 20 otherwise the QT will be equal to 20. This may lead to poor waiting time and turnaround time if processes' burst are higher than 20.

Al-Bakhrani et al., (2020) compared the performance of the four basic CPU scheduling algorithms – FCFS, SJF, RR and priority scheduling for educational purpose and established that SJF gives better performance in terms of average waiting time and turnaround time.

Sohrawordi et al. (2019) proposed a modified RR which provide a dynamic quantum time in each cycle. In each cycle, averages of processes' burst are used as QT and processes are sorted according to their burst size before scheduling.

Biswas et al. (2018); Biswas and Samsuddoha (2019) proposed an improved RR with a fix QT based on maximum difference between two adjacent processes. The QT is the absolute sum of the maximum difference between the adjacent processes and the sum of the least processes' burst within the cycle. Processes are sorted in increasing order where shorter process are executed first before the larger ones.

SLU JST
Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Zouaoui et al. (2019) proposed a priority based RR CPU scheduling where processes are assigned priority number and are executed based on their priorities.
Mody and Mirkar (2019) proposed a Smart RR CPU scheduling algorithm with dynamic QT. The QT is calculated as the sum of the average difference between adjacent processes' bursts with half of the average difference between adjacent processes' bursts. Processes are sorted, and the shorter processes are assigned CPU first. If the remaining process burst is greater than or less than the QT, the burst time will be its QT otherwise it will be assigned the calculated QT and return the process to the ready queue.
Srujana et al. (2019) tried overcoming the challenges of the classical RR algorithm by combining the product of SJF and RR algorithms to minimize the waiting and response time of a process. This was achieved by assigning an amount of execution time to a process and implemented using the least burst time first. The result of the combination improved the latency, minimized waiting time and starvation.
An improved scheduling algorithm was proposed by Aijaz et al. (2019) known as Efficient Round Robin Algorithm (ERRA). The improvement was based on the classical round-robin (RR) where the quantum time (QT) was made to be dynamic as opposed to the constant QT in classical RR. The results of the proposed algorithm (ERRA) were better in terms of average response time, average waiting time (AWT) and the context switching (CS)
Mishra and Rashid (2014) proposed a dynamic RR where processes are assigned CPU after they have been sorted in ascending order. In the first round, the QT is the least burst time for processes in the ready queue. In the next cycle the least burst time will be the next QT and so on.

5.1 Pitfalls Identified from the Related Works

Despite efforts to enhance the classical Round Robin (RR) algorithm in the reviewed literatures, two primary pitfalls have been identified:

1. **Calculation of Quantum Time (QT):** None of the reviewed studies provided a standardized method for determining the quantum time (QT) in classical RR. Instead, an arbitrary small value is often assigned as QT. This approach can lead to excessive context switching if the QT is too small, consequently increasing waiting and turnaround times.

2. **Use of Sorting Techniques:** Several studies applied sorting techniques to minimize waiting and turnaround times by allowing processes with shorter burst times to execute first and potentially run to completion in RR. However, evaluating proposed models using sorted processes against classical RR with unsorted processes may unfairly favor the sorted approach. It's well-known that RR with sorted processes performs better than RR with unsorted processes, thus skewing comparisons.

6. Proposed Method

In the proposed method – Dynamic Adjacent Quantum Time Round Robin (DAQRR), the quantum time for each process in the ready queue is different. The variable quantum time allocated to processes make the algorithm to be dynamic and each process can have at most two quantum times – their adjacent (left and right) processes bursts. The maximum between the adjacent processes' bursts will be assign as the QT for that process. Processes are assumed to be in the ready queue as they appear without been sorted. The pseudocode and the flowchart are presented in algorithms 1 and 2 below respectively:

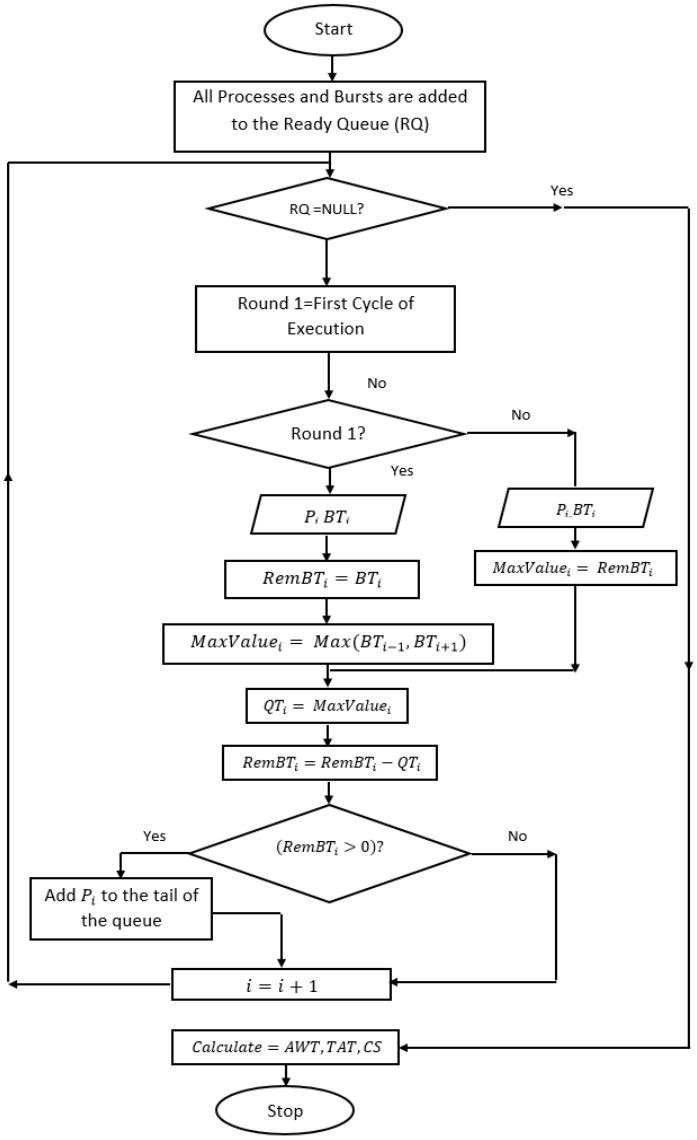
Algorithm 1: Pseudocode for the proposed methodology

1: **Start Procedure**
2: *Processes arrive the ready queue*
3: **Do**
4: **For** each process P_i
5: **If** *P_i is the first execution cycle (Round 1)*

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

6: *Get the adjacent bursts time for P_i (left-Burst and right-Burst)*
7: *Assign the maximum between the left-Burst and right-Burst as the quantum time for P_i for that cycle*
8: *Calculate P_i remaining burst time $RemBT_i$*
9: *Update the remaining burst time $RemBT_i$*
10: **Else** *(not Round 1)*
11: *Assign remaining burst time, $RemBT_i$ as Quantum time for P_i*
12: *Add $RemBT_i$ to the end of the queue for next round execution*
13: **Loop While** *(ready queue is not empty)*
14: *Calculate the Average waiting time (AWT), Average turnaround time (TAT) and number of context switching (CS).*
15: **End Procedure**

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng



Algorithm 2: Flowchart for the proposed methodology

7. Illustration and Algorithm Analysis

While assuming that processes are already in the ready queue, the following toy data are considered for illustration as shown in Table 1.

Table 1: Processes and their burst time in the ready queue.

Processes (P)	Burst Time (BT)
P1	12
P2	34
P3	8
P4	19

The execution pattern for the processes can be presented as follows:

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

P4→ P1 → P2→ P3→ P4→ P1

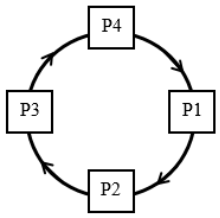


Figure 2: Circular process arrangement

In figure 2 above, the adjacent processes for P1 is P4 and P2, for P2 is P1 and P3, for P3 is P2 and P4 while that of P4 is P3 and P1.

Therefore, in round 1:
P1 quantum time = maximum between P4 and P2
P2 quantum time = maximum between P1 and P3
P3 quantum time = maximum between P2 and P4
P4 quantum time = maximum between P3 and P1

While in round 2:
Any remaining process P_i with burst time BT_i will have its quantum time to be BT_i
P1 whose burst time is 12 will have 34 time unit as its quantum time since its adjacent processes' bursts are 19 and 34 and the maximum between them is 34 time unit. Likewise, P2 whose burst time is 34 will have 12 time unit as its quantum time since its adjacent bursts are 12 and 8 and the maximum between them is 12. P3 and P4 will have quantum time of 34 and 12 time unit respectively. As P1 and P3 run to completion, P2 and P4 will be preempted for the next cycle with remaining bursts time of 22 and 7 time unit respectively. In the second round, each process will run to completion using its remaining burst time as its quantum time. That is, in the second cycle, the two processes P2 and P4 with 22 and 7 burst time respectively will run to completion.

7.1. Proposed Algorithm (DAQRR) Example

Using table 1 above, the Gantt chart is generated in figure 3.

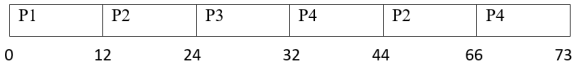


Figure 3: DAQRR Gantt chart

P1=0
P2=12+20=32ms
P3=24ms
P4=32+22=54ms
AWT=0+32+24+54=110/4=27.5ms
ATAT= (0+12) + (32+34) + (24+8) + (54+19) =12+66+32+73=183/4=45.75ms

7.2. Classical Round Robin Example

To be fair to classical RR, quantum time is given as the average of processes' bursts in the ready. Using table 1, the quantum time will be:
Quantum time=12+34+8+ 19 =73/4=18.25≈18
And the Gantt chart is represented in figure 4.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

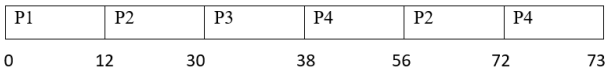


Figure 4: Classical RR Gantt chart

P1=0ms
P2=12+26=38ms
P3=30ms
P4=38+16=54ms
AWT=0+38+30+54=122/4=30.5ms
ATAT= (0+12) + (38+34) + (30+8) + (54+19) =12+72+38+73=195/4=48.75ms

8. Algorithm Implementation and Testing

The implementation assumes all processes are initially in the ready queue with an arrival time of zero and no assigned priority. Random variables within specified upper and lower bounds are generated to represent burst times for these processes. The implementation is carried out using Python programming language, utilizing three parameters to generate the bursts.

The simulation is conducted across 10 iterations, starting with 10 processes and increasing up to 100 processes. Key metrics such as average waiting time, average turnaround time, and number of context switches are computed. These metrics serve as the basis for comparing the performance of the classical Round Robin (RR) algorithm against the proposed RR variant. The outcomes of these tests are graphically represented for clarity and comparison.

```
Enter the number of proces: 7
Enter the lower bound: 10
Enter the upperbound: 90
Process burst: [60, 65, 73, 46, 33, 54, 17]
```

Figure 5: Input for the system and the process bursts

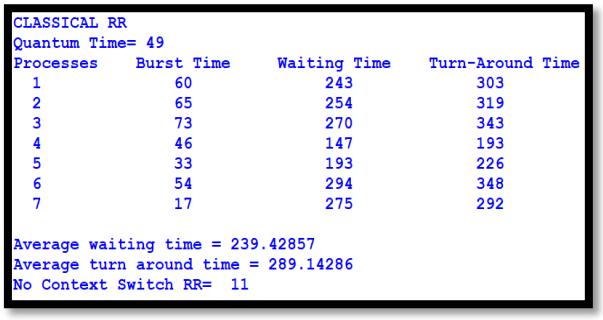
```
CLASSICAL RR
Quantum Time= 49
Processes    Burst Time     Waiting Time     Turn-Around Time
  1              60             243              303
  2              65             254              319
  3              73             270              343
  4              46             147              193
  5              33             193              226
  6              54             294              348
  7              17             275              292

Average waiting time = 239.42857
Average turn around time = 289.14286
No Context Switch RR=  11
```

Figure 6: Simulation output for classical RR

Figure 6 above shows the average waiting time, average turnaround time and number of context switching using classical RR technique for the processes' bursts in figure 5.

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng



```
CLASSICAL RR
Quantum Time= 49
Processes    Burst Time     Waiting Time    Turn-Around Time
  1             60             243             303
  2             65             254             319
  3             73             270             343
  4             46             147             193
  5             33             193             226
  6             54             294             348
  7             17             275             292

Average waiting time = 239.42857
Average turn around time = 289.14286
No Context Switch RR=  11
```

Figure 7: Simulation output for DAQRR

Figure 7 above shows the average waiting time, average turnaround time and number of context switching using the proposed technique for the processes' bursts in figure 5.
The results from figures 6 and 7 shows that the proposed technique has significant improvement against the classical RR in terms of average waiting time, average turnaround time and number of context switching.

9. Results Presentation and Discussion

Table 2 presents the simulation results from running the program 10 times with varying numbers of processes (NoP): 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. For each run, lower bounds (LB) and upper bounds (UB) are specified as inputs to the system, randomly generated to determine process bursts.
The average waiting time (AWT), average turnaround time (ATAT), and number of context switches (CS) are calculated for both the classical Round Robin (RR) and Dynamic Adjacent Quantum Time Round Robin (DAQRR). Across all scenarios, DAQRR consistently minimizes AWT, ATAT, and CS compared to classical RR. For example, with 10 processes and LB and UB set to 10 and 100 respectively, the classical RR AWT is 439.8, ATAT is 510.3, with 16 CS, whereas DAQRR achieves AWT of 392.6, ATAT of 463.1, and 14 CS. This represents a 5.7% reduction in AWT and a 4.9% reduction in ATAT with DAQRR.
The performance advantages of DAQRR persist as the number of processes increases to 60, 70, 80, and 90, as indicated in the table.

Table 2: Simulation results between Classical RR and DAQRR

SIMULATION RESULT BETWEEN CLASSICAL RR AND DAQRR								
NoP	**LB**	**UB**	**Classical RR**			**DAQRR**		
			AWT	**ATAT**	**CS**	**AWT**	**ATAT**	**CS**
10	10	100	439.8	510.3	16	392.6	463.1	14
20	24	97	747.7	810.55	30	704.35	767.2	27
30	35	243	3584.175	3726.275	59	3497.1	3639.2	54
40	25	89	1398.925	1454.5	60	1282.125	1337.7	53
50	30	143	3071.78	3163.04	76	2740.7	2831.96	67
60	23	89	2349.2	2406.617	90	2092.567	2149.983	79
70	7	176	4514.843	4612.271	108	3984.114	4081.543	95
80	12	234	6400.463	6522.463	124	5588.788	5710.788	106
90	34	300	9589.533	9754.256	137	8753.489	8918.211	119
100	52	465	16741.4	17000.67	149	15703.33	15962.6	133

Where,

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

NoP represents number of process
LB represents lower bound used to generate random variables
UB represents upper bound used to generate random variables

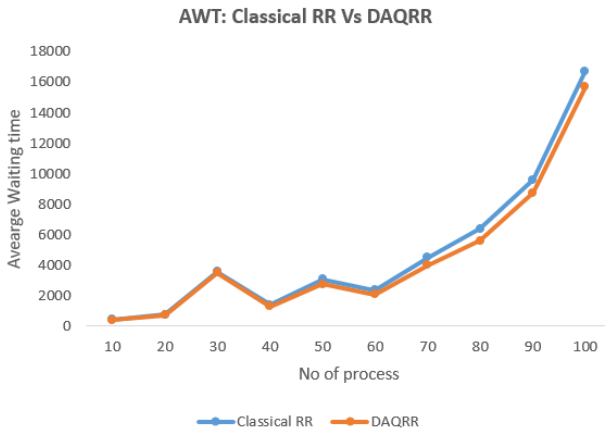


Figure 8: AWT for Classical RR versus DAQRR

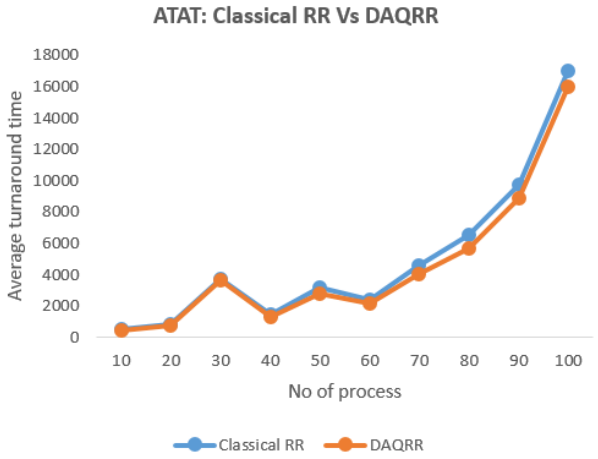


Figure 9: ATAT for Classical RR versus DAQRR

Sule Lamido University Journal of Science & Technology Vol. 8 No. 2 [Dec, 2024], pp. 83-95
https://doi.org/10.56471/
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

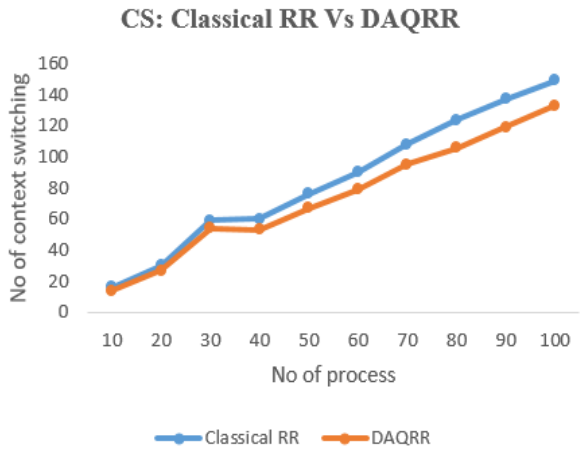


Figure 10: Number of context switching for Classical RR versus DAQRR

Figures 8, 9 and 10 respectively shows that, given any number of process and choice of lower and upper bound, the proposed algorithm (DAQRR) performed better by minimizing average waiting time, average turnaround time and number of context switching.

10. Conclusion

The results presented clearly demonstrate that Round Robin CPU scheduling can be significantly enhanced through the implementation of a dynamic quantum time approach, as exemplified by the proposed algorithm—Dynamic Adjacent Quantum Round Robin CPU Scheduling (DAQRR). DAQRR adapts the quantum time dynamically by utilizing the maximum adjacent processes' bursts as the quantum time for allocating CPU time to processes in the ready queue.

Comparison with classical Round Robin shows that the proposed DAQRR algorithm consistently reduces average waiting time, average turnaround time, and number of context switches. These improvements underscore the effectiveness of dynamic quantum time in optimizing CPU scheduling performance.

References

Abubakar, S. E., Yusuf, S. A., Obiniyi, A. A., & Mohammed, A. (2023). Modified Round Robin with Highest Response Ratio Next CPU Scheduling Algorithm using Dynamic Time Quantum. *Sule Lamido University Journal of Science & Technology 6*(2), pp.87-99. doi: doi.org/10.56471/slujst.v6i.363

Ahmed, W. F, & Muquit, S. P. (2016). Improved Round Robin Scheduling Algorithm with Best Possible Time Quantum and Comparison and Analysis with the RR Algorithm. *International Research Journal of Engineering and Technology (IRJET), Vol. 3(3)*, pp. 1357-1361.

Aijaz, M., Tariq, R., Ghori, M., Rizvi, S. W., & Qazi, F. (2019). *Efficient Round Robin Algorithm (ERRA) using the Average Burst Time.* Paper presented at the 2019 International Conference on Information Science and Communication Technology (ICISCT), Karachi, Pakistan. 9 -10 March, 2019.

Al-Bakhrani, A., Hagar, A., Hamoud, A., & Kawathekar, S. (2020). Comparative Analysis of CPU Scheduling Algorithms: Simulation and Its Applications. *International Journal of Advanced Science and Technology, Vol. 29(3)*, pp. 483-494.

Ali, S. M., Alshahrani, R. F., Hadadi, A. H., Alghamdi, T. A., Almuhsin, F. H., & El-Sharawy, E. E. (2021). A Review on the CPU Scheduling Algorithms: Comparative Study. *International Journal of Network Security, Vol. 21(1)*, pp.19-26. doi: 10.22937/IJCSNS.2021.21.1.4

Biswas, D., Saha, S., Faisal, H., & Samsuddoha, M. (2018). An Improved Round Robin Scheduling Algorithm based on Maximum Difference of Two Adjacent Processes. *Barishal University Journal Part 1, Vol. 5(1&2)*, pp. 257-271.

Biswas, D., & Samsuddoha, M. (2019). Determining Proficient Time Quantum to Improve the Performance of Round Robin

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Scheduling Algorithm. *International Journal of Modern Education and Computer Science, 11*, 33-40. doi: 10.5815/ijmecs.2019.10.04

Danjuma, S., Dams, G. L., and Simon, A. (2021). *Proposed Approach for Resource Allocation Management in Service Oriented Architecture (SOA) Environment.* Science World Journal. **Vol. 16 (No 3)**: p. pp. 357-362.

Farooq, M. U., Shakoor, A., & Siddique, A. B. (2017). *An Efficient Dynamic Round Robin algorithm for CPU scheduling.* Paper presented at the 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), Islamabad, Pakistan. 8-9 March, 2017.

Fiad, A., Mekkakia M. Z.., & Bendoukha, H. (2020). Improved Version of Round Robin Scheduling Algorithm Based on Analytic Model. *International Journal of Networked and Distributed Computing, Vol. 8(4)*, pp. 195-202. doi: 10.2991/ijndc.k.200804.001

Harki, N., Ahmed, A., & Haji, L. (2020). CPU scheduling techniques: A review on novel approaches strategy and performance assessment. *Journal of Applied Science and Technology Trends, Vol. 1(2)*, pp. 48-55.

Indusree, J., & Prabadevi, B. (2017). Enhanced round robin CPU scheduling with burst time based time quantum. *IOP Conference Series: Materials Science and Engineering, Vol. 263 (4)*, pp. 1-8. doi: 10.1088/1757-899X/263/4/042038

Iqbal, M., Ullah, Z., Khan, I. A., Aslam, S., Shaheer, H., Humayon, M., Salahuddin, M. A. Mehmood, A. (2023). Optimizing Task Execution: The Impact of Dynamic Time Quantum and Priorities on Round Robin Scheduling *Future Internet 15*(104), pp. 1-14. doi: doi.org/10.3390/fi15030104

Jbara, Yosef Hasan. (2019). *A new Improved Round Robin-Based Scheduling Algorithm-A comparative Analysis.* Paper presented at the 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia. 3-4 April, 2019

Joshi, A., & Goyal, S. B. (2019). *Comparison of Various Round Robin Scheduling Algorithms.* Paper presented at the 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India. 22-23 Nov. 2019

Majedkan, N. A., Ahmed, A. J., & Haji, L. M. (2020). CPU Scheduling Techniques: A Review on Novel Approaches Strategy and Performance Assessment. *Journal of Applied Science and Technology Trends, Vol. 1(2)*, pp. 48 –55. doi: doi: 10.38094/jastt1215

Mishra, M, & Rashid, F. (2014). An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum. *International Journal of Computer Science, Engineering and Applications, Vol. 4(4)*, pp. 1-8. doi: 10.5121/ijcsea.2014.4401

Mody, S. & Mirkar, S. (2019). *Smart Round Robin CPU Scheduling Algorithm For Operating Systems.* Paper presented at the 2019 4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT).

Mora, H., Abdullahi, S. E., & Junaidu, S. B. (2017). *Modified Median Round Robin Algorithm (MMRRA).* Paper presented at the 2017 13th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria. 28-29 Nov. 2017.

Moseley, B. & Vardi, S. (2022). The Efficiency-fairness Balance of Round Robin Scheduling. *Operations Research Letters, 50*(1), pp. 20-27. doi: 10.1016/j.orl.2021.11.008

Paul, T., Faisal, R., & Samsuddoha, M. (2019). Improved Round Robin Scheduling Algorithm with Progressive Time Quantum. *International Journal of Computer Applications, 178*, 30-36. doi: 10.5120/ijca2019919419

Sakshi, S., Chetan, S., Shamneesh, S., Sandeep K., Shami A., Khalil, E.M, & Ali W. (2022). A new Median-Average Round Robin Scheduling Algorithm: An Optimal Approach for Reducing Turnaround and Waiting Time. *Alexandria Engineering Journal, 61*(12), pp. 10527-10538. doi: doi.org/10.1016/j.aej.2022.04.006

Seemakuthi, S., Siriki, V. A., & Lydia, E. L. (2015). A Review on Various Scheduling Algorithms. *International Journal of Scientific & Engineering Research, Vol. 6(12)*, pp. 769-779.

Shafi, U., Shah, M., Wahid, A., Abbasi, K., Javaid, Q., Ashghar M. & Haider, M. (2020). A Novel Amended Dynamic Round Robin Scheduling Algorithm for Timeshared Systems. *The International Arab Journal of Information Technology, Vol. 17(1)*, pp. 90-98. doi: 10.34028/iajit/17/1/11

Silberschatz, A., Galvin, P. B., & Gagne, G. (2005). *Operating System Concepts* (7th ed.). USA: John Wiley and Sons.

Simon, A., Abdullahi, S., & Junaidu, S. (2014b). Half Life Variable Quantum Time Round Robin (HLVQTRR). *International Journal of Computer Science and Information Technologies, Vol. 5(6)*, pp. 7210-7217.

Simon, A., Dams, G. L. and Danjuma, S. (2022). *An Improved Half Life Variable Quantum Time with Mean Time Slice Round Robin CPU Scheduling (ImHLVQTRR).* Science World Journal. **Vol. 17(No 2)**: p. pp. 248-253.

Simon, A., Saleh, A., & Junaidu, S. (2014a). Dynamic Round Robin with Controlled Preemption (DRRCP). *International Journal of Computer Science Issues, Vol. 11(3)*, pp. 109-117.

Singh, M., & Agrawal, R. (2017). *Modified Round Robin Algorithm (MRR).* Paper presented at the 2017 IEEE International

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India. 21-22 Sept. 2017.

Sohrawordi, M., Ali, E., Uddin, P., & Hossain, M. (2019). A Modified Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum. *International Journal of Advanced Research (IJAR), Vol. 7(2)*, pp. 422-429. doi: 10.21474/IJAR01/8506

Srujana, R., Roopa, Y. M., & Mohan, M. D. S. K. (2019). *Sorted Round Robin Algorithm.* Paper presented at the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India. 23-25 April, 2019.

Vayadande, K., Bodhankar, A., Mahajan, A., Prasad, D., Dhakalkar, R., & Mahajan, S.. (2024). *An Improved Way to Implement Round Robin Scheduling Algorithm*. Paper presented at the High Performance Computing, Smart Devices and Networks, Singapore.

Zouaoui, S., Boussaid, L., & Abdellatif, M.. (2019). Priority based round robin (PBRR) CPU scheduling algorithm. *International Journal of Electrical and Computer Engineering (IJECE), Vol. 9(1)*, pp. 190-202. doi: 10.11591/ijece.v9i1.pp190-202

Zouaoui, S., Boussaid, L., & Abdellatif, M. (2016). *CPU scheduling algorithms: Case & comparative study.* Paper presented at the 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Sousse, Tunisia. 19-21 Dec., 2016.