Journal of Education and Science (ISSN 1812-125X), Vol: 31, No: 04, 2022 (105-115)

Improved Round Robin CPU Scheduling Algorithm with Different Arrival Times Based on Dynamic Quantum

Abdulnasir Younis Ahmad[1]*

[1*]Department of Computer Science, Education College for Pure Science, University of Mosul, Mosul, Iraq

E-mail: [1*] abdulnasir.younus@uomosul.edu.iq

(Received September 04, 2022; Accepted October 31, 2022; Available online December 01, 2022)

DOI: 10.33899/edusj.2022.135082.1273, © 2022, College of Education for Pure Science, University of Mosul.

Abstract

Modern operating systems are based on the principle of time-sharing in executing simultaneous operations. Determining the length of the time slice, and the times when processes arrive at the ready queue are problems that affect metrics as the average waiting time (AWT), average turnaround time (ATAT), response time (RT) and the number of context switches (NCS) of the time-sharing round robin RR algorithms. The research aims to propose an algorithm that achieves a short waiting time while maintaining a reasonable response time, which is the most important characteristic of time-sharing algorithms. The Different Arrival-Dynamic Quantum Round Robin (DADQRR) algorithm bases its work on different parameters to adjust the time slice value dynamically. The algorithm has been compared to three other algorithms that are similar in terms of dealing with different arrival times, namely AN, MARR, RR. The algorithm outperformed the three algorithms at range from 6.155% to 31.409% in term of AWT. It achieved an outperformance of 5.924% to 30.850%, considering the TAT. The ranges of outperformance values resulted from the difference in the ranges of arrival times, as well as in the ranges of burst times.

Keyword: Dynamic quantum, varying arrival times, Round Robin, turnaround time, waiting time

خوارزمية راوند روبن محسنة باوقات وصول مختلفة ومرتكزة على شريحة زمنية ديناميكية

عبدالناصر يونس احمد[1]*

[1*]قسم علوم الحاسوب، كلية التربية للعلوم الصرفة، جامعة الموصل، الموصل، العراق

الخلاصة:

تعتمد أنظمة التشغيل الحديثة على مبدأ مشاركة الوقت في تنفيذ العمليات المتزامنة. يبرز تحديد طول الشريحة الزمنية ولحظة وصول العمليات الى طابور الجاهزية كمشكلة تؤثر في متوسط زمن الانتظار (AWT) ومتوسط زمن المكوث (ATAT) وزمن الاستجابة (RT)وعدد مرات تبديل السياق (NCS) في خوارزميات هذه الانظمة. يهدف البحث الوصول الى خوارزمية تحقق زمن انتظار قصيرمع المحافظة على زمن استجابة معقول وهو الخصيصة الاهم في خوارزميات مشاركة الوقت. ترتكز خوارزمية Different Arrival-Dynamic Quantum Round Robin (DADQRR) في عملها على معلمات عديدة لضبط قيمة الشريحة الزمنية ديناميكيًا. تمت مقارنة الخوارزمية بثلاث خوارزميات أخرى تشبهها من حيث التعامل مع أوقات وصول مختلفة هي AN, MARR, RR. تفوقت الخوارزمية على الخوارزميات الثلاث باعتبار متوسط زمن الانتظار حيث حققت تفوقا يتراوح قدره بين 6.155% و 31.409% . وحققت تفوقا يتراوح قدره بين 5.924% و 30.850% باعتبار متوسط زمن امكوث. ان الاختلاف في قيم التفوق نتج عن الختلاف في مديات ازمان الوصول وكذلك في مديات ازمان التنفيذ.

الكلمات المفتاحية : شريحة زمنية ديناميكية، اوقات وصول مختلفة، راوند روبن، زمن المكوث ، زمن الانتظار

1. Introduction

Today, software systems control all modern devices. The operating system (OS) is a software program that runs on computers and other devices to offer a suitable environment for each user to implement advanced programs[1]. It provides a user interface that enables computer hardware and user programs to perform their functions correctly and appropriately. The operating system controls how hardware operates, how data and information are input and output, and how user programs are interfaced. The Operating system is the software that performs and implements scheduling of processes.[2]

When talking about the operating system, terminology like multitasking and multiprocessing are utilized. These terms can be used interchangeably. Multiprocessing refers to using many CPUs in processing[3]. Even though only one CPU is being used for processing, because it switches between tasks so quickly, the user thinks that all apps are active at once. Multitasking is the term for it [4],[5],[3].

Numerous processes can run simultaneously on a system and share and in turn maximizes resource consumption. These systems compete for execution among a large number of processes running in memory, and are known as time-sharing systems. As a result, CPU time must be distributed among all of these processes[6].

Preemptive and non-preemptive process execution methods are two types of process execution methods. Non-preemptive processes run in sequential order, which means one process runs at a time while other processes wait for the previous process to finish. The term "preemptive process execution" refers to the allocation of CPU resources when a process is ready to run.

A scheduler is a program that selects the tasks to be scheduled according to a specified algorithm [1]. The CPU scheduler is responsible for allocating resources, specifically CPU time, to all processes.[2] There are three sorts of schedulers: long-term, mid-term, and short-term. The first is a long-scheduler, which determines the process that moves from the process pool on a hard disk to Random Access Memory (RAM) (HDD) [1]. The short-term scheduler chooses one of the jobs that are ready to run and assigns CPU time to it. To limit the degree of multiprogramming, the mid-term scheduler eliminates less used or idle processes from memory. The process of selecting a particular process at a particular moment to give it the CPU and for a specified period is called scheduling. The scheduling algorithm affects the behavior of the system. As a result, it has an impact on its performance. [7]

An operating system can choose a process and send it to the CPU for execution in many ways. Every scheduling algorithm has its own set of benefits and drawbacks[8],[9]. The system's behavior must be optimized, based on a specific criterion by choosing the best scheduling algorithm for a specific class of processes [6].

A scheduling mechanism known as First Come First Served (FCFS) adds newly incoming processes to the end of the queue and CPU is assigned to the head processes for execution. According to their arrival time(AT), the initial processes are assigned to the CPU. The average waiting time(AWT) of this algorithm is quite high. [8],[9].

Another scheduling algorithm is the Shortest Job First (SJF) scheduling algorithm, which assigns the CPU to the task that has the shortest CPU burst time first. The short-term scheduler pushes the processes with the shortest CPU burst time to the front of the queue and inserts the processes has the longest CPU burst time to the back. The biggest issue with this scheduling is determining when the next CPU burst will occur [10].

In Round Robin (RR) scheduling algorithm, each process is assigned to the CPU for a certain time slice (quantum). This prevents starvation since each process is given the same amount of time to complete. The main flaw in this approach is the context switch. If the quantum is little, more context switches will occur, resulting in low CPU efficiency, and if it is high, the algorithm will act like FCFS [11].. Time quantum usually lasts between 10 and 100 milliseconds[12], and 80 percent of CPU bursts should, as a general rule, be less than the time quantum[13].

For CPU scheduling, throughput is a crucial metric. It refers to the number of processes accomplished per unit of time. Another metric is TAT which is the amount of time between submitting a process and

having it judged. The CPU scheduling technique only has an impact on a metric called waiting time (WT) which is the overall period of time spent in the ready queue. Another measurement is the time it takes to begin reacting is referred to as response time(RT) [14]. A Context switch (CS) is a measurement which is the practice of conserving the state of an active process and resuming the state of a preempted process to allow for the continuation of an active process's execution from the same place at a later time. Context switching is typically computationally demanding, resulting in time and memory waste. The scheduler and operating system frequently incur this cost[7] [15],[14]

There are many distinct CPU-scheduling algorithms, each with its own set of attributes, and selecting the right one can favor one class of activities over another. The features of the various algorithms must be examined while deciding which algorithm to utilize in a certain situation. [16]

High scheduling performance may result in a situation in which increasing one trend undermines performance in another[17], [6].

The main attributes of Round Robin (RR) algorithm are:

1. Process starvation may be prevented since each process has a turn.
2. When compared to other methods, processes with shorter burst times benefit from early execution.
3. RR algorithm is anticipatory. [18]

In this research, the performance of the Round Robin algorithm has been improved by improving the values of waiting time, turnaround time and the number of context switch operations. That was reached by devising an algorithm for dynamic adjustment of the time quantum value, taking into account the arrival times to ready queue as well as the throughput of the CPU.

Sections of this paper are as follows. Section 2 presents how the research papers related to our work deals with the problem. In section 3, the way of handling arriving processes was presented, how the time quantum was calculated and the factors that affected that calculation. Results were presented in section 4 and conclusions were provided in section 5.

2. Related works

[19] offer a new algorithm, termed AN, built upon a novel strategy known as a dynamic time-quantum. This tactic's objective is to have operating systems alter time quantum in response to the burst time (BT) of a group of awaiting processes in the ready queue. The algorithm deals with processes that arrive at different ATs. The examples presented in the paper present a small number of processes and with values of BTs and ATs that soon change the algorithm to an algorithm with fixed ATs.

[20]use Dynamic Time Quantum to build Improved Round Robin (IRR) Scheduling. The ready queue is first reconfigured with respect to the minimum BT. The average of the median of all operations and the longest BT is then given as the best time quantum according to their algorithm. The answer is then applied to all operations as the optimal time quantum. On each cycle of execution, the same procedure is applied for the quantum time until all processes have completed their tasks. Results showed that the suggested IRR results in fewer context shifts, a lower AWT, and a lower ATT than the conventional Round Robin.

Optimized Round Robin with Dynamic Time Quantum is developed by[21]. The quantum time is calculated by dividing the total of all BTs in the queue by the number of processes in the ready queue (RQ). Processes that are unable to complete their execution after the first phase of processing are removed from the RQ since their BT exceeds the quantum. The mean BT of processes in the second phase serves as the quantum for that phase. The same reasoning is employed up until one process remains in a phase, at which point the BT of that process is automatically taken as the quantum time. Using NCS, AWT, and ATT criteria, the suggested approach outperforms the algorithms SARR, MRR, RR, RP-5, IRRVQ, and DQRRR. In comparison to other methods, the suggested technique savings are 41% of AWT and 31% of ATT. The NCS and ART were not considered by the author.

[22] develop an Efficient Round Robin CPU Scheduling Algorithm. The quantum time is calculated using two methods: first, the median of all BTs is determined and multiplied by the maximum BT. Second, the BT mean is calculated and multiplied by the shortest BT. The square root is determined

as the quantum time after both outcomes are added together. The resulting result is compared to traditional RR and two alternative techniques. Using WT, context switch, and TAT as metrics of evaluation, the suggested approach surpasses all previous algorithms. The proposed algorithm's main flaw is that its RT was high when compared to other algorithms.

With an unknown BT, [23] offer a DTSRR scheduling algorithm. This algorithm shows that the queue is attended to using a dynamic time quantum for all processes. Based on FCFS on the arrival queue, all processes are attended. For the second cycle, the original quantum is doubled for the processes that are unable to accomplish their jobs. When comparing the regular RR with an optimized RR, their findings suggest that the turnaround is reduced by about 15%, the WT is reduced by 15%, and the context switch is reduced by around 10%.

In [24], the researchers developed a novel algorithm, which consists of a number of algorithms that have been combined to create a single algorithm. The goal is to reduce the waiting time. It also decreases the number of switches to create an efficient algorithm. The new algorithm enhanced productivity depending on the obtained results.

To overcome the drawbacks of the Round-Robin approach and lower RT and WT in processes with a predetermined amount of runtime, the researchers in [26] merged the Shortest Job First and Round-Robin algorithms. The research discovered that the rated RR is appropriate for application since it reduced RT and WT while also lessening starvation by giving the shortest burst duration priority.

In a novel strategy proposed by [24], the time quantum varies cycle by cycle according to the amount of time left in the process. A Process has the choice to take a more quantum time when it needs a small additional time to finish its execution. The Smart Round Robin is the result of the two working together. Compared to the Traditional Round Robin Scheduling approach, it results in lower AWT and lower TAT.

[11] proposed the SIDRR Scheduling Algorithm. The BT values of all the processes in the queue are multiplied to produce the quantum, which is then computed by taking the nth root of the product. This quantum time is used in the suggested method. The nth root is determined by how many processes are in the queue. AWT, AVT, and NCS are used to test the suggested algorithm. The suggested approach is also compared to the NIRR, DABRR, IRRVQ, RMRR, and EDRR algorithms. When compared to previous algorithms, the result reveals an improvement in performance. When compared to the previous approach, the suggested technique saves 49% in terms of WT. The proposed algorithm was not evaluated using ART.

A Modified Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum is presented by[9]. The quantum used is the average burst time(ABT). The results of the experiment reveal that the proposed algorithm outperforms the traditional RR, and RRVQ algorithms since it requires less WT and TAT. In comparison to RRVQ, the suggested method reduced ATT by 10.8 percent and AWT by 12 percent.

Use of the enhanced method Median-Average Round Robin(MARR) is advised by the authors of [3]. The author suggests a dynamic time quantum for the system using the median and the ABTs of the processes. The proposed model was compared with four more scheduling techniques by the authors. They clearly show that their suggested algorithm produces better results with a smaller ATT and WT.

3. Methodology

The algorithm adopts the idea of entering processes in the queue in real time. That is, the process enters the competition for the CPU time as soon as it arrives and thus is based on different access times in the ready queue.

The process that arrives first in the queue is handled to be executed. Processes that arrive during execution are added to the queue instantly, after which the process that its incomplete is inserted. The next process is then extracted from the ready queue to be executed in the same way and so on. Execution continues in this way until the ready queue stops receiving new processes.

A quantum time is allocated to each process that pops from the front of the ready queue for execution. The value of this quantum is determined by a certain algorithm. After the end of the quantum, the remaining burst time (RT) is considered and the process is given an additional time if the time required to complete execution is less than or equal to half the value of the quantum. This procedure reduces the number of context switches as well as the process's WT.

A variable AccBT is defined to keep the cumulative sum of BTs for processes when they enter the queue, and another variable IQPN for the number of processes in the queue. For every process that enters the queue, its BT is added to AccBT, and the number of processes in the queue IQPN is also incremented. The executed time of a process is subtracted from ACCBT and IQPN is decremented when the process is completed. The value of IQNP remains the same in a situation where the process is not completed and returns to the queue.

A variable CWOQ to count the number of processes completed in one cycle is defined, and another one RTRQ to count the processes returning to the queue waiting for another cycle. The flow chart that depicts the proposed (DADQRR) algorithm is shown in figure (1)

Time Slice

The value of the quantum depends mainly on a value equal to the cumulative BTs of the processes in the RQ divided by their number. The relationships between the completed processes and the processes completed in one turn and that return to the queue with the number of processes in the queue are used to modify the value of the quantum when executing each process.

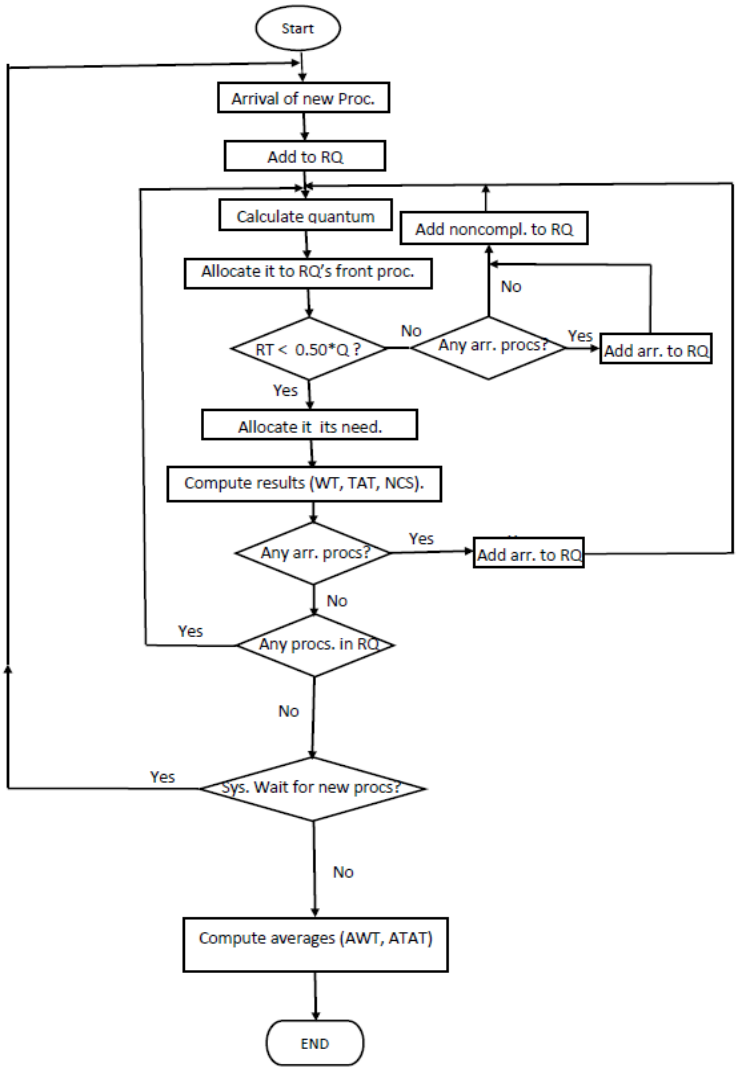


Figure 1. Flowchart of proposed algorithm

The quantum is increased according to specified conditions by the equation:
$Q = AccumBT/(IQPN - 1)$ ..... (1)
And it is decreased according to specified conditions by the equation:
$Q = AccumBT/(IQPN + 1)$ ..... (2)

Where:
Q : Quantum
$AccumBT$: Accumulated BTs
$IQPN$: In queue processes number

This method of adjusting the quantum prevents the algorithm from falling into execution in FCFS manner, especially in cases where many processes in front of the ready queue have relatively short BTs and a few processes with long BTs are at the tail of ready queue. The quantum adjusted to be greater than BTs of nearly 70% of the processes. At the same time, it also prevents the occurrence of high frequency of context switching and thus creates an additional high load on the system. The algorithm of calculating the quantum is shown in figure (2)

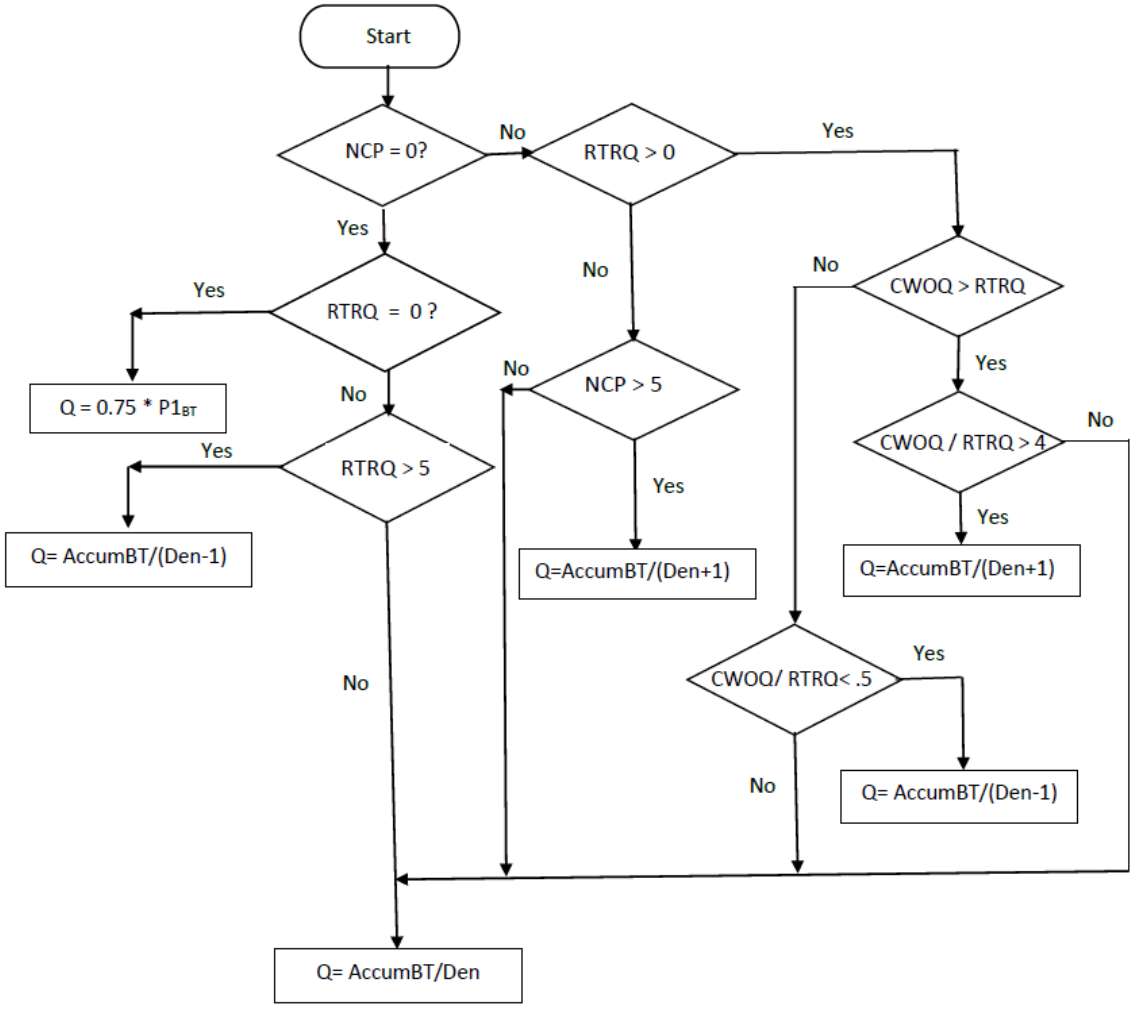


Figure 2. Flowchart for calculating quantum

The value of the initial time slice was determined according to the relationship:
Q=0.75 BT
This quantum is granted for the first process in the queue. This process is excluded from granting additional time.

4. Results and Discussion

Data preparation

The data meant here are BTs and ATs. The data used can be described through the following aspects:
1.Value: The data values have a significant impact on the results that are considered a criterion for comparison (such as WT, TAT, CS …etc.).
So in order to avoid the possibility of choosing values of specific BTs with specific ATs to give results that are also specific, the data were generated randomly and with a regular distribution. The RANDBETWEEN () function in the MS Excel application was used to achieve this purpose.
2. Range:
Burst times: Two ranges of BTs were used. The first is with BTs in the range of 2 to 10 time units, and the second with a range equal to four times the first, i.e. 2 to 40 time units.
Arrival times: The range of ATs and the number of processes indicates the rate at which processes flow into the queue. Two ranges of ATs were selected, the first with 0 to 20 and the second with 0 to 40.
The Value interacts with the Range in determining the level of the processes backlog in the queue.
3- The number of processes: The number of processes plays a major role in the accuracy of measuring the relative performance of the algorithm. The small number of data prevents the algorithm from presenting its effect in dealing with the data, and therefore it is not possible to obtain accurate results to base on for comparison.
In order to compare the performance of the proposed algorithm with other algorithms, the following cases have been suggested:
Case 1:
The algorithms AN[19], MARR[3], and DADQRR were executed, as well as the traditional round robin algorithm with time slices of 2 TRR-2[13], and 5 TRR-5[13], with BTs in range 2-10. Figures (3), (4), (5) show the resultant AWTs, ATTs, and the NCSs respectively.

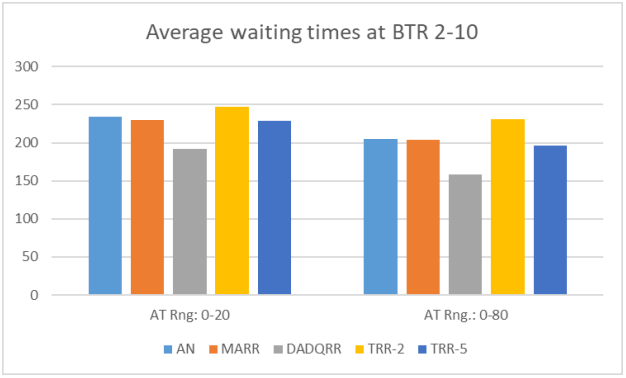


Figure 3. Average waiting times at BT range 2-10

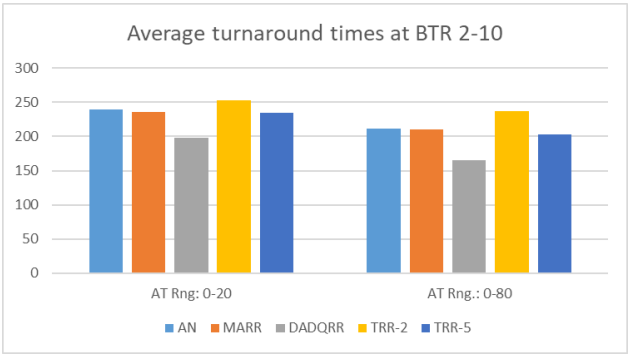


Figure 4. Average turnaround times at BT range 2-10

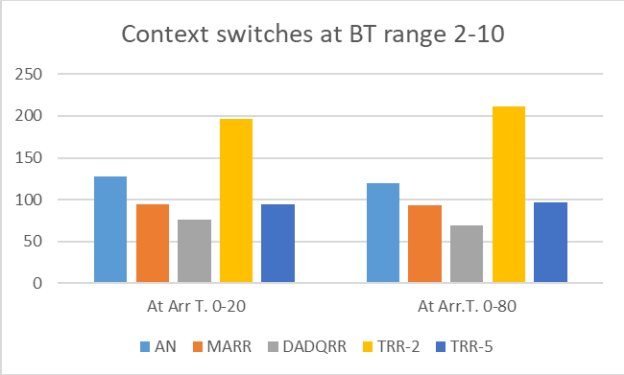


Figure 5. Context switches at BT range 2-10

Case 2:

The algorithms AN, MARR, and DADQRR were executed, as well as the traditional round robin algorithm with slices of 10, 15, 20, and 30 and with BTs in range 2-40. Figures (6), (7), (8) show the resultant AWTs, ATTs, and the NCSs respectively:

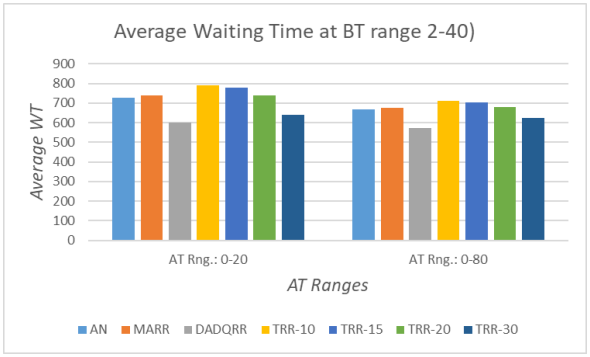


Figure 6. Average waiting times at BT range 2-40

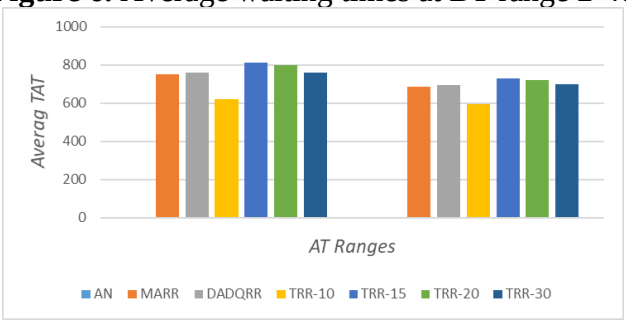


Figure 7. Average turnaround times at BT range 2-40

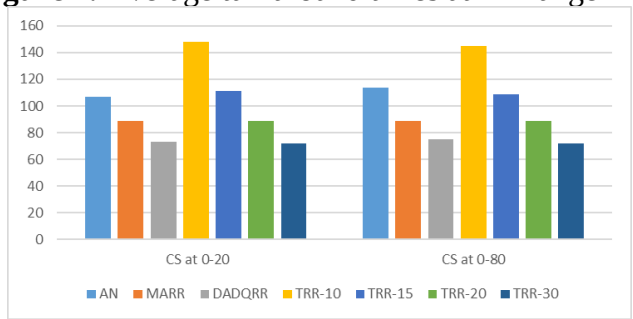
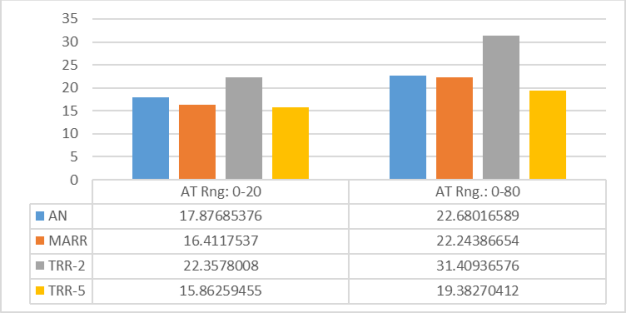


Figure 8. Context switches at BT range 2-40

In order to evaluate the performance of the proposed algorithm, it has been implemented and compared with other algorithms that have been designed to work with different access times. Three algorithms MARR, AN, and TRR with different time slices were selected for comparison as following:

Case 1:

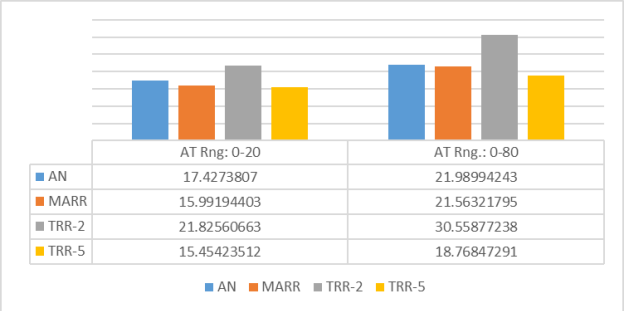
In case 1, in AWT at AT Range of 0-20 and BT range 2-10 DADQRR outperforms algorithms AN, MARR by %17.877 and %16.412 respectively. It also outperforms TRR-2 and TRR-5 by %22.358 and %15.862 respectively. At AT range of 0-80, the outperformances of the supposed algorithm are shown in figure (9).



	AT Rng: 0-20	AT Rng.: 0-80
AN	17.87685376	22.68016589
MARR	16.4117537	22.24386654
TRR-2	22.3578008	31.40936576
TRR-5	15.86259455	19.38270412

Figure 9. The ratios of AWT outperformance of DADQRR over other algorithms

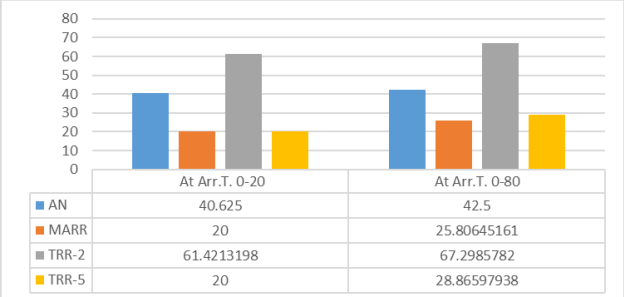
Figures 10 and 11 show TAT and CS outperformance ratios of DADQRR over other algorithms at BT range of 2-10



	AT Rng: 0-20	AT Rng.: 0-80
AN	17.4273807	21.98994243
MARR	15.99194403	21.56321795
TRR-2	21.82560663	30.55877238
TRR-5	15.45423512	18.76847291

AN MARR TRR-2 TRR-5

Figure 10. The ratios of TAT outperformance of DADQRR over other algorithms

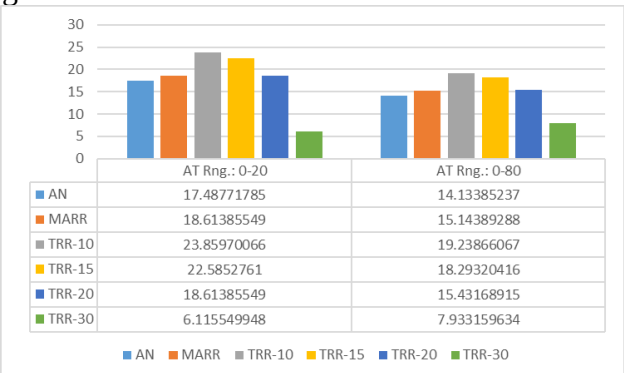


	At Arr.T. 0-20	At Arr.T. 0-80
AN	40.625	42.5
MARR	20	25.80645161
TRR-2	61.4213198	67.2985782
TRR-5	20	28.86597938

Figure 11. The ratios of context switches outperformance of DADQRR over other algorithms

Case 2:

Figures 12, 13, 14 show AWT, TAT and CS percentage ratios of outperformances of DADQRR over other algorithms at BT range of 2-40.



	AT Rng.: 0-20	AT Rng.: 0-80
AN	17.48771785	14.13385237
MARR	18.61385549	15.14389288
TRR-10	23.85970066	19.23866067
TRR-15	22.5852761	18.29320416
TRR-20	18.61385549	15.43168915
TRR-30	6.115549948	7.933159634

AN MARR TRR-10 TRR-15 TRR-20 TRR-30

Figure 12. The ratios of AWT outperformance of DADQRR over other algorithms

The only minor/small underperformance shown is in the number of CS in comparison with TRR-30.

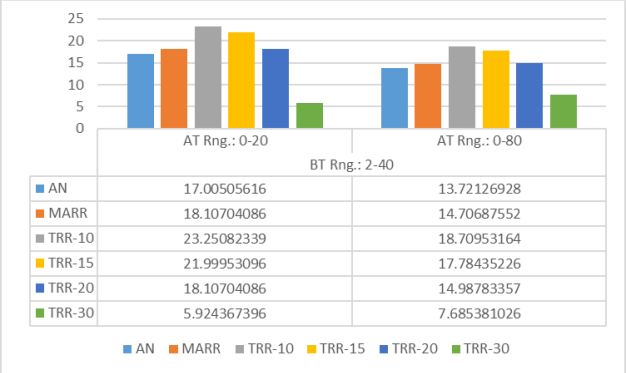


Figure 13. The ratios of TAT outperformance of DADQRR over other algorithms

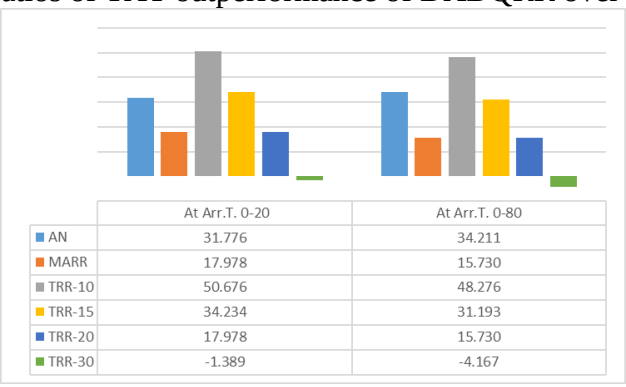


Figure 14. The ratios of Context switches outperformance of DADQRR over other algorithms

5. Conclusion

The proposed algorithm benefits from adjusting the quantum considering the relation between the number of completed processes, completed processes with one turn and the number of returned processes to the ready queue. Additionally, the suggested DADQRR algorithm allows processes that are near to completion of their execution to finish it and move on, which reduces the number of processes in the RQ and the NCSs. DADQRR algorithm was compared with three common algorithms from the point of view of AWT, TAT, and the NCSs. The proposed algorithm outperformed all the others. However, it underperformed only one algorithm quite a little in the number of context switches in one case.

6. Acknowledgements

The authors would like to thank the University of Mosul / Education College for Pure Science for their facilities, which have helped to enhance the quality of this work.

7. References

[1] N. Harki, A. Ahmed, and L. Haji, "CPU Scheduling Techniques: A Review on Novel Approaches Strategy and Performance Assessment," *J. Appl. Sci. Technol. Trends*, vol. 1, no. 2, pp. 48–55, 2020, doi: 10.38094/jastt1215.

[2] D. Biswas and M. Samsuddoha, "Determining Proficient Time Quantum to Improve the Performance of Round Robin Scheduling Algorithm," *Int. J. Mod. Educ. Comput. Sci.*, vol. 11, no. 10, pp. 33–40, 2019, doi: 10.5815/ijmecs.2019.10.04.

[3] Sakshi *et al.*, "A new median-average round Robin scheduling algorithm: An optimal approach for reducing turnaround and waiting time," *Alexandria Eng. J.*, vol. 61, no. 12, pp. 10527–10538, 2022, doi: 10.1016/j.aej.2022.04.006.

[4] S. Zouaoui, L. Boussaid, and A. Mtibaa, "Priority based round robin (PBRR) CPU scheduling algorithm," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 1, p. 190, 2019, doi: 10.11591/ijece.v9i1.pp190-202.

[5] T. Paul, R. Hossain, and M. Samsuddoha, "Improved Round Robin Scheduling Algorithm with Progressive Time Quantum," *Int. J. Comput. Appl.*, vol. 178, no. 49, pp. 30–36, 2019, doi:

10.5120/ijca2019919419.
[6] S. M. Mostafa, "Proportional Weighted Round Robin: A Proportional Share CPU Scheduler in Time Sharing Systems," *Int. J. New Comput. Archit. their Appl.*, vol. 8, no. 3, pp. 142–147, 2018, doi: 10.17781/p002491.
[7] S. Mody and S. Mirkar, "Smart Round Robin CPU Scheduling Algorithm for Operating Systems," *4th Int. Conf. Electr. Electron. Commun. Comput. Technol. Optim. Tech. ICEECCOT 2019*, no. December 2019, pp. 309–316, 2019, doi: 10.1109/ICEECCOT46775.2019.9114602.
[8] K. I. Arif, M. Morad, M. A. Mohammed, and M. A. Subhi, "AN EFFICIENT THRESHOLD ROUND-ROBIN SCHEME for CPU SCHEDULING (ETRR)," *J. Eng. Sci. Technol.*, vol. 15, no. 6, pp. 4048–4060, 2020.
[9] S. a, "a Modified Round Robin Cpu Scheduling Algorithm With Dynamic Time Quantum.," *Int. J. Adv. Res.*, vol. 7, no. 2, pp. 422–429, 2019, doi: 10.21474/ijar01/8506.
[10] U. Shafi *et al.*, "A novel amended dynamic round robin scheduling algorithm for timeshared systems," *Int. Arab J. Inf. Technol.*, vol. 17, no. 1, pp. 90–98, 2020, doi: 10.34028/iajit/17/1/11.
[11] T. O. Omotehinwa, S. I. Azeez, and S. S. Olofintuyi, "A Simplified Improved Dynamic Round Robin (SIDRR) CPU Scheduling Algorithm," *Int. J. Inf. Process. Commun.*, vol. 7, no. 2, pp. 122–140, 2019, [Online]. Available: https://ijipc.com.ng/index.php/ijipc/article/view/301/176.
[12] A. Igbaoreto, E. Oyekanmi, and T. O. Omotehinwa, "An Improved Round Robin CPU Scheduling Algorithm for Asymmetrically Distributed Burst Times," vol. 1, no. October, pp. 50–68, 2019.
[13] Abraham Silberschatz, "Operating System Concpets," *Wiley*. pp. 3–54, 2018.
[14] N. Mittal, K. Garg, and A. Ameria, "A paper on modified round robin algorithm," *Int. J. Latest Technol. Eng. Manag. Appl. Sci.*, vol. 4, no. 11, pp. 93–98, 2015.
[15] E. Oyetunji and A. Oluleye, "Performance Assessment of Some CPU Scheduling Algorithms," *Res. J. Inf. Technol.*, vol. 1, no. 1, pp. 22–26, 2009, [Online]. Available: http://www.airitilibrary.com/Publication/alDetailedMesh?docid=20413114-200907-201008160029-201008160029-22-26.
[16] S. Mostafa and H. Amano, "An Adjustable Round Robin Scheduling Algorithm in Interactive Systems," *Inf. Eng. Express*, vol. 5, no. 1, pp. 11–18, 2019, doi: 10.52731/iee.v5.i1.353.
[17] S. M. Mostafa, S. Z. Rida, and S. H. Hamad, "Finding Time Quantum of Round Robin Cpu Scheduling Algorithm in General Computing Systems Using Integer Programming," *Spring*, vol. 5, no. October, pp. 64–71, 2010.
[18] J. R. Indusree and B. Prabadevi, "Enhanced round robin CPU scheduling with burst time based time quantum," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 263, no. 4, 2017, doi: 10.1088/1757-899X/263/4/042038.
[19] A. Noon, A. Kalakech, and S. Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average," no. June, 2011, [Online]. Available: http://arxiv.org/abs/1111.5348.
[20] D. Nayak, S. Kumar Malla, and D. Debadarshini, "Improved Round Robin Scheduling using Dynamic Time Quantum," *Int. J. Comput. Appl.*, vol. 38, no. 5, pp. 34–38, 2012, doi: 10.5120/4607-6816.
[21] A. R. Dash, S. kumar Sahu, and S. K. Samantra, "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum," *Int. J. Comput. Sci. Eng. Inf. Technol.*, vol. 5, no. 1, pp. 07–26, 2015, doi: 10.5121/ijcseit.2015.5102.
[22] G. Siva Nageswara Rao and S. V. N. Srinivasu, "An efficient round robin cpuscheduling algorithm using dynamic time slice," *Int. J. Pharm. Technol.*, vol. 8, no. 4, pp. 21461–21469, 2016.
[23] A. Muraleedharan, N. Antony, and R. Nandakumar, "Dynamic time slice Round Robin scheduling algorithm with unknown burst time," *Indian J. Sci. Technol.*, vol. 9, no. 8, 2016, doi: 10.17485/ijst/2016/v9i8/76368.
[24] S. Mody and S. Mirkar, "Smart Round Robin CPU Scheduling Algorithm for Operating Systems," *4th Int. Conf. Electr. Electron. Commun. Comput. Technol. Optim. Tech. ICEECCOT 2019*, pp. 309–316, 2019, doi: 10.1109/ICEECCOT46775.2019.9114602.