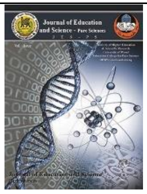
Journal of Education and Science (ISSN 1812-125X)

www.edusj.mosuljournals.com



An Attempt to Set Standards for Studying and Comparing the Efficiency of Round Robin Algorithms

A. Y. Ahmad

Department of Computer Science, Education College for Pure Science, University of Mosul, Mosul, Iraq

Article information

Article history:
Received :January 12, 2023
Accepted :March 05, 2023
Available online: June 01, 2023

Keywords:
Dynamic time quantum
Performance comparison
Scheduling criteria
Round Robin
Waiting time

Correspondence:
A. Y. Ahmad
abdulnasir.younus@uomosul.edu.iq

Abstract

With the advent of the need for interactive systems, the urgent need for time-sharing systems has emerged. Round-robin algorithms have emerged to achieve time-sharing. The performance of time-sharing systems depends largely on the length of the time slice in the round-robin algorithms. The time slice length affects the measuring criteria of the performance of the algorithms. Researchers suggested and are continuing to suggest algorithms to obtain the best values for the time slice. Adopting one algorithm over another in a system and for a class of applications requires choosing the best-performing algorithm. This research is an attempt to develop an objective approach for accurate comparison between algorithms. For the sake of objectivity in comparison, five algorithms similar in their general characteristics were chosen; a Modified Median Round Robin Algorithm (MMRRA), A New Median-Average Round Robin Scheduling Algorithm (NMARR), An Improved Round Robin Scheduling Algorithm with Varying Time Quantum (IRRVQ), A Modified Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum (RRDTQ), Improved Round Robin Algorithm with Progressive Dynamic Quantum (IRRPDQ). The results showed that the outperformance of an algorithm over a group of algorithms according to a specific criterion is not absolute, permanent, and fixed in value and that resorting to statistical measures is the best way to clarify the degree of performance of the algorithms.

DOI: 10.33899/edusj.2023.137735.1317, ©Authors, 2023, College of Education for Pure Sciences, University of Mosul.

Introduction

An operating system is a type of system software used to control computer hardware and offer services for application programs; it is built to enable users to communicate with the hardware. Additionally, operating systems offer a platform on which other apps, such as Chrome, Microsoft Office, games, etc., can run [1].
The CPU scheduling algorithm determines which tasks should be completed first and which tasks should be completed last to achieve maximum CPU utilization. The fundamental CPU scheduling algorithms include Round Robin(RR), Priority Scheduling, Shortest Job First (SJF), and First Come First Serve (FCFS)[1].
In RR scheduling, each process receives a time quantum, which is a portion of CPU time [2]. This time quantum plays a key role in regulating the performance of the system[3]. It is crucial to RR because if it is too small, RR behaves like a Process Sharing algorithm and results in more context switches. Conversely, if it is sufficiently large, RR behaves like FCFS[2]. Round Robin algorithm suffers from a long waiting time, an enormous amount of context switches, and low throughput at certain positions[4].
Every scheduling job based on unique characteristics to select the specified scheduling algorithm. One or more of the following criteria must be taken into account to achieve the desired goal [4].

- CPU utilization: A scheduling method must deliver efficient and ideal performance. To do this, it is necessary to create and use an algorithm to exploit CPU time and optimize CPU utilization.
- Context switches: As context switching is considered a system overhead, the number of context switches in a scheduling algorithm should be minimal. Context switches need to be reduced to improve CPU utilization and boost algorithm performance.
- Throughput: By throughput, we mean the total number of processes that are finished in a certain amount of time. Therefore, if an algorithm achieves higher throughput, it can be said that it is a good scheduling algorithm.
- Turnaround time: The scheduling algorithm with the shortest turnaround time is preferable. By increasing CPU utilization, the shorter turnaround time will enhance the performance.
- Waiting time: Some processes must wait for their turn to operate when others are running on the CPU. Processes' waiting times should be reduced because they have an impact on the system's overall effectiveness.

Because of the continuous change in the remaining burst time values of the processes, the fixed time quantum does not give the best results. Thus, many proposed algorithms used a dynamic time quantum approach to enhance the performance[3]. There is a need to define accurately and objectively which algorithm optimizes a specified criterion. The research is an attempt to develop an objective approach for accurate comparison between algorithms.

Sections of this paper are as follows. Section two shows how researchers compared and evaluate their proposed algorithms. Section three presents the factors that affect the scheduling criteria and how the time quantum was calculated in several algorithms. Results were presented in section four and conclusions were provided in section five.

Related Works

In [5], the suggested algorithm An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum outperformed the traditional RR approach in terms of performance. The suggested IRRVQ scheduling algorithm decreased waiting and turnaround times, and thus enhanced system performance. The algorithm was compared considering two sets of processes. The first consists of 5 processes, with burst times in the range of 10-32, at an average of 21-time units for each process. Processes were arranged in ascending order of burst times with a fixed arrival time for all of them equal to zero. The second consists of 5 processes, with an arrival time in the range of 5-36, an average of 18-time units for each process. The processes were arranged according to their arrival times. The comparison was made with the traditional RR, with time slices of 10, 6, and 5 for the first group, and 11 and 7 for the second group.

[3] presented an improved dynamic round robin algorithm which is based on a dynamic time quantum. The algorithm was tested on three sets of processes; The first consists of 7 processes arranged in ascending order, with execution times in the range of 20-120, and with an average of 60-time units. The second is of 5 processes arranged according to their arrival times, with an average burst time of 40-time units. And the third is of 7 processes arranged in descending order according to burst time, with burst times in the range of 5-80 time units, at a rate of 31.5-time units for each process. The algorithm was compared with two other algorithms, one was round robin, with a time slice of 40, 25, and 20 for the three groups, respectively. Traditional Round Robin and Self Adjustment Round Robin were compared with the suggested algorithm. Due to shorter average waiting and turnaround times, the experimental findings demonstrate that the suggested algorithm is superior to Round Robin and Self Adjustment Round Robin. Round Robin and SARR, however, performed better than the suggested algorithm when the burst times of the processes were ordered decreasingly.

[6] declared that their proposed algorithm had reduced average waiting time, average turnaround time, and the number of context switches. The algorithm was applied to 3 groups of processes; The first consists of 5 processes arranged in ascending order of burst time, with burst times in the range 14-77 with an average of 46.5-time units per process and a fixed arrival time of zero. The second consists of 5 processes that were arranged in ascending order, with a burst time in the range of 22-74 at a rate of 49.4-time units for each process, and an arrival time in the range of 0-9, with an average arrival time of one process per 5.75-time units. The third consists of 4 processes arranged randomly according to the arrival time with burst times in the range of 15-85, with an average burst time of 51.75-time units, and arrival times in the range of 0-20, with an arrival rate of one process per 13-time units. The algorithm provided better performance than simple RR. Researchers explained that when the quantum had been increased and a process with a small burst time was to arrive in the middle of execution then the algorithm could suffer - the new one has to wait more time than the basic RR.

[7] proposed an algorithm that had improved performance. The algorithm was tested on one set of 5 processes, with execution times in the range of 26-82, with an average execution time of 50-time units. With a fixed access time of zero, the algorithm was compared with five other algorithms. Among the algorithms is the traditional round robin with a time slice of 25-time units. The algorithm considered arranging operations in ascending order according to the execution time. In comparison to the RR method, there were a lot fewer context switch. Additionally, it shortened turnaround and average waiting times.

[8] announced that their Modified Median Round Robin Algorithm (MMRRA) was tested on five processes with burst times in the range of 45–73-time units and compared with five other algorithms. When it was compared to the traditional RR, IRR, IMRRSJF, HLVQTRR, and DRRCP CPU scheduling algorithms, it offered a more effective solution. MMRRA dramatically reduces the number of context switches (NCS), but the average waiting time and turnaround time were not greatly impacted.
[9] presented Smart Round Robin which outperformed Traditional Round Robin and other algorithms in terms of AWT and ATAT. First, the algorithm executes processes with a short remaining burst time and provides a dynamic time quantum for each cycle. The algorithm was tested on four sets of processes; The first consists of four processes with a large burst time in a range of 8-34 and an average of 18-time units for each process, the second of five processes with a short burst time in a range of 2-9 and an average of 5-time units for each process, the third of four processes with burst times in a wide range of 11-82 time units and an average of 46.5-time units for each process, and the fourth of four processes in a range of 5-11, at a rate of 7.5-time units per process. The arrival times for all processes in the first three groups were 0, while the fourth group had different arrival times in a range of 0-7, with an average of one process arriving every three-time units. The algorithm was compared with the traditional round robin algorithm with time slices of 6, 4, 20, and 2-time units for the four groups, respectively. The algorithm considered arranging processes in ascending order according to the burst time.
In [10], a comparison of calculations was performed on two groups of processes; The first consists of 6 processes, with a burst time in the range of 4-8, at a rate of 6-time units for each process, and an arrival time in a range of 0-13. The second consists of 5 processes with burst times in the range of 7-90 and arrival times in the range of 0-10. The algorithm considered adding the processes to the ready queue as soon as it arrives, but it chooses the shortest one when allocating the CPU to the next process. The algorithm had been compared to Dynamic Quantum Using the Mean Average Round Robin (ANRR), Shortest Remaining Burst RR (SRBRR), An Optimized Round Robin Algorithm (ORR), Adaptive Round Robin Algorithm (ARR), and Simple Round Robin Algorithm (RR) algorithms. The comparative analysis demonstrated that, in terms of average waiting time and average turnaround time, the suggested method outperforms the mentioned algorithm.
[11] presented a new CPU scheduling algorithm with varying dynamic time quantum. Two groups were considered in the test and comparison. The first group consists of five processes with burst times in a range of 10-32 and the second group of five processes also in a range of 10-35 with different arrival times. The processes were arranged in ascending order according to their burst times. The algorithm was compared to the traditional round robin (RR), AN, and IRRVQ. The comparison demonstrated that the suggested RRDTQQ algorithm achieved a lower average waiting time, a lower turnaround time, and a smaller number of context switches.
[12] Described a novel CPU scheduling strategy called An Improved Time Varying Round Robin Algorithm (ITVRR). The method outperformed the traditional RR algorithm. The algorithm was tested on one set of seven processes, with burst times in a range of 7–58-time units, an average burst time of 8.5-time units, and arrival times in a range of 0–6. The algorithm handled the processes in the order of arriving at the ready queue. For CPU scheduling, the ITVRR algorithm was compared against the FCFS, SJF, RR, and RMRR algorithms. According to the results, ITVRR outperformed RR and RMRR algorithms in terms of AVT and CS, although RMRR algorithms only exceeded the suggested model in terms of AWT. Additionally, ITVRR outperformed FCFS in terms of AVT.

In [13], a CPU scheduling algorithm called ADRR Scheduling is suggested. The dynamism of time quantum and many rounds, which produced optimal waiting times and numbers of context switches, were some of the key characteristics of ADRR. The comparison is made by applying four examples, each of them on five processes. The burst times of the processes were in the range of 5-22, 10-60-, 17-50-, 4-10-, and 5–35-time units. Other well-known scheduling techniques were compared to the algorithm. The findings demonstrate that the suggested ADRR algorithm outperformed competing algorithms in terms of decreased turnaround times, fewer context switches, and decreased average waiting times.
[4] presented a new Median-Average Round Robin (MARR) scheduling algorithm. The algorithm was tested on three sets of processes. The first included 4 processes with execution times in the range of 6-80, the second included 8 processes in a range of 10-200, and the third included 6 processes in a range of 12-140 with fixed arrival times in each of the three groups. The algorithm adopted the method of entering operations into the ready queue in the form of successive groups. A group is not entered until after the completion of the group before it. The algorithm adopted arranging operations in ascending order according to the burst time in the ready queue. An average turnaround time and average waiting time are decreased using the suggested algorithm.

[1] presented A novel intelligent RR Algorithm. For a large number of processes, clustering the ready queue and building sub-ready queues based on optimal threshold values and standard deviation had been suggested. The algorithm was applied for comparison on one set of processes with burst times in a range of 2-300 and with one fixed arrival of zero. The algorithm adopted the ascending order of processes according to their burst times. Experimental results demonstrated that the suggested algorithm performed better in terms of average waiting time (AWT), average turnaround time (AWT), and context switches (NCS).

Methodology

Round Robin algorithms are used in time-sharing systems and interactive systems because they provide fast responses to the user and prevent starvation. As mentioned previously, the performance of Round Robin algorithms depends largely on the value of the time slice. Many time-sharing algorithms have been proposed and modified, all aiming to improve one or more scheduling criteria such as AWT and ATAT, and the number of context switches. The influencing aspects of these factors are:

- The arrangement of processes in the ready queue
- time slice value
- Burst time values
- arrival times
- Method of handling processes

- The arrangement of processes in the ready queue

In general, the researchers adopted two methods of arranging the processes in the ready queue before start allocating CPU for them. The first is to arrange them in ascending order, considering the burst time, meaning that the process with the shortest burst time is at the front of the queue. This method is adopted because the implementation of short processes at the beginning inevitably reduces the amount of AWT. The second is to consider arrival times, priorities, or other things, which results in a random arrangement of burst tines. This method was not widely adopted because the results for the same values of burst times were different.

- The time quantum
  The time quantum affects the performance of the round robin algorithm scheduling algorithm. In a traditional round robin, a specific time slice value is imposed. This value must consider the range of burst time values, as long burst times require a large quantum, and vice versa. With developed algorithms, a variable time slice value is used depending on the execution times of the processes in the ready queue.
- Burst Time Values
  Large burst times increase AWT and ATAT values in single CPU systems. The effect of burst times on these values is reduced by choosing the perfect time slice or increasing the number of processors or cores in the system.
- Arrival times
  Some algorithms adopt a single arrival time for all processes, while others adopt different arrival times. Arriving at the same time gives the algorithm freedom in the ordering of operations, while the second approach imposes a random order arrangement of burst times.
- Entering processes into the ready queue
  When ready processes arrive, they are dealt with in one of two ways; The first is to be received in a waiting queue that precedes the ready queue. From the waiting queue, it is moved as a group to the ready queue. The second is to receive a process directly into the ready queue and allow it to compete for CPU with the processes in the ready queue.

For a fair comparison among a group of algorithms, algorithms must be chosen with similar basic characteristics. In this paper, the chosen algorithms were IRRVQ [5], MMRRA [8], IRRPDQ [10], RRDTQQ[11], and NMARR[4]. The similar characteristics of algorithms that were considered are:

- Processes were dealt with as a group all of its processes must be carried out so that another group can start executing.
- Fixed arrival times (with a value of zero) for all processes.
- Ascending order management of processes according to the value of the burst time.
  The main difference that characterizes the algorithms is how the time quantum is calculated. The following is how the time quantum in each algorithm was calculated:

1. An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum (IRRVQ)[5]
(Time quantum)$_{ri}$= Minimum burst time value of the processes in the ready queue
2. Modified Median Round Robin Algorithm (MMRRA) [8]
(Time quantum)$_{ri}$= Square root of (median value of the sorted burst times * highest burst time value)
3. Improved Round Robin Scheduling Algorithm with Progressive Time Quantum (IRRPDQ) [10]
If the number of processes is less than 4 then
(Time quantum)$_{ri}$= (Average of burst time of processes in the ready queue)$_{ri}$
If the number of processes is even, then

(Time quantum) $_{ri}$= (Bt_{front} + Bt_{tail} + $Bt_{(QLength/2)}$ + $Bt_{(QLength/2 + 1)}$)/4
If the number of processes is odd, then
(Time quantum) $_{ri}$= (Bt_{front} + Bt_{tail} + $Bt_{ceil(QLength/2-1)}$ + $Bt_{ceil(QLength/2 + 1)}$)/4
Where:
$Bt_{front:}$ Bust time of the process at front of the ready queue
Bt_{tail}: Burst time of the process at the tail of the ready queue
$Bt_{(QLength/2)}$: Burst time of the process at location (ready queue length/2)
$Bt_{(QLength/2 + 1)}$: Burst time of the process at location((ready queue length/2)+1)
4. A modified Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum (RRDTQ)[11]
(Time quantum) $_{ri}$= (Average of burst time of processes in the ready queue) $_{ri}$
5. A New Median Average Round Robin(NMARR) [4]
(Time quantum) $_{ri}$= (($MBtRq_{ri}$ + $AvBtRq_{ri}$) / 2)

Where:
ri: specified round i
MBtRq: Median of burst time values of processes in the ready queue
AvBtRq: Average of burst time values of processes in the ready queue
The above-mentioned algorithms were coded and tested by the author applying the examples mentioned in the research papers. The results were compared to ensure that algorithms were coded correctly.

Results and discussion

To be certain that an algorithm is superior to a certain criterion over another algorithm or group of algorithms, this algorithm must outperform them when applied to several cases within different ranges considering a specified parameter. In this paper, a comparison was first made among three algorithms; NMARR[4], RRDTQ[11], and IRRPDQ [10], then two more algorithms were added; MMRRA [8], and IRRVQ [5]. The same data was used to make the comparison between the five algorithms.
The algorithms are implemented on groups whose process's burst times fall within three ranges. Each range contains ten groups of processes, and each group contains ten processes whose burst time values have been randomly generated. That is, the execution was repeated ten times in each range.
In burst times at a range of 5-20, the result showed that, at one time, IRRPDQ was the best and RRDTQ was the worst, at another time, RRDTQ was the best and IRRPDQ was the worst as shown in Table 1 and Figure 1.
The average for average waiting times for the ten groups of Algorithm NMARR is superior to that of Algorithm RRDTQ by 0.071 and that of Algorithm IRRPDQ by 0.059. Table 1.

Table 1. Average waiting times at burst time range of 5-20

Burst Time Range 5-20											
Method	Grp1	Grp2	Grp3	Grp4	Grp5	Grp6	Grp7	Grp8	Grp9	Grp10	Average
NMARR	56.2	77.1	41.1	65.8	42.8	67.8	64.1	70.2	54.3	77.3	61.67
RRDTQ	61.2	76.1	48.5	79.6	43.4	73.6	63.1	83.4	59.3	76.3	66.45
IRRPDQ	51.6	76.1	38.8	79.6	43.4	73.6	64.1	83.4	59.3	83.3	65.32

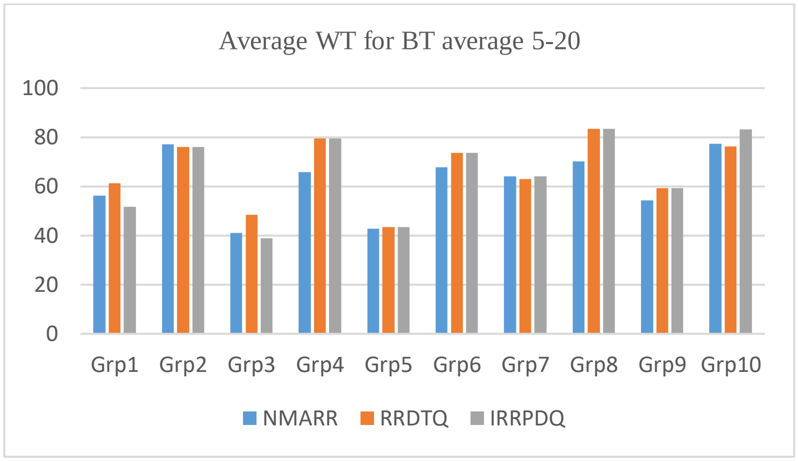


Figure 1. Average waiting times at burst time range of 5-20

In the burst times range of 5-50, the average of average waiting times of the 10 groups of Algorithm IRRPDQ outperformed that of Algorithm RRDTQ by 0.0016, and that of Algorithm NMARR by 0.0073. Table 2 and Figure 2.

Table 2. Average waiting times at burst time range of 5-50

Burst Time Range 5-50											
Method	Grp1	Grp2	Grp3	Grp4	Grp5	Grp6	Grp7	Grp8	Grp9	Grp10	Average
NMARR	152.8	111.8	116.2	116.3	169.2	118.2	131.7	114.6	90.1	126.2	124.71
RRDTQ	149.8	110.8	117.4	116.3	164.7	118.8	130.7	115.2	90.1	126.2	124
IRRPDQ	151.3	111.8	116.2	126.9	164.7	117	111	114	99.5	125.6	123.8

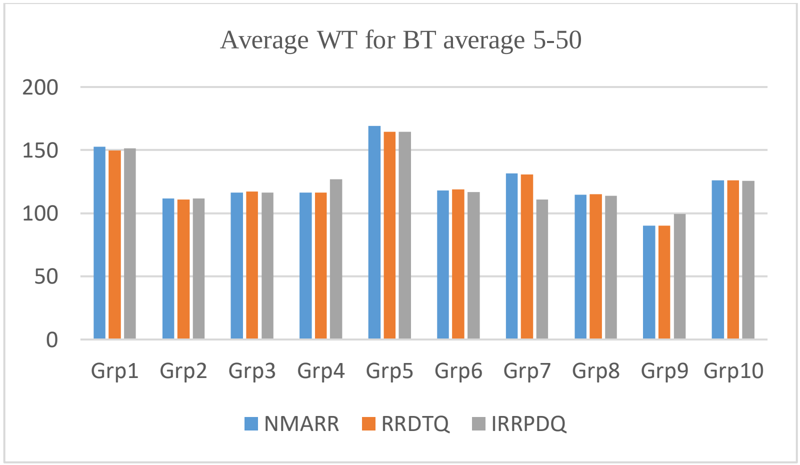


Figure 2. Average waiting times at burst time range of 5-50

In the burst times range of 5-100, the average for average waiting time values of the 10 groups of Algorithm NMARR outperformed that of Algorithm RRDTQ by 0.0033, and that of Algorithm IRRPDQ by 0.0053. Table 3 and Figure 3.

Table 3. Average waiting times for burst time range of 5-100

Burst Time Range 5-100											
Method	Grp1	Grp2	Grp3	Grp4	Grp5	Grp6	Grp7	Grp8	Grp9	Grp10	Average
NMARR	319	341.3	213.4	216.2	240.3	232.1	214.7	145.6	209.9	120.8	225.33
RRDTQ	348.5	333.8	216.4	220.2	239.3	233.3	196.3	147.4	203.9	121.8	226.09
IRRPDQ	319	332.3	213.4	215.2	240.3	252.3	213.7	148	208.4	122.8	226.54

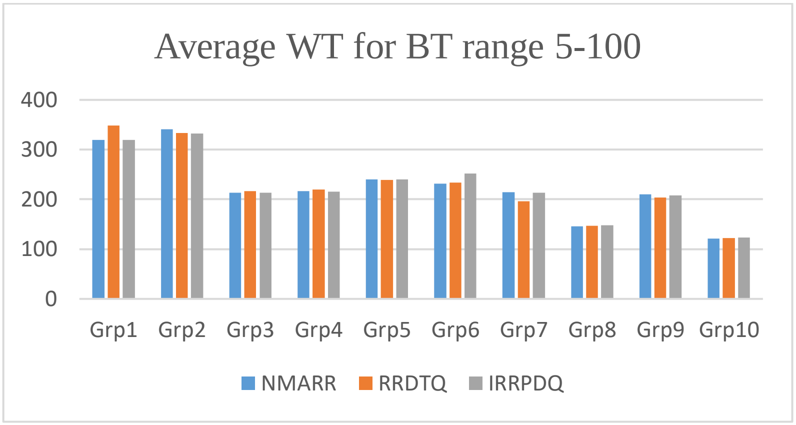


Figure 3. Average waiting times at burst time range of 5-100

By adding algorithms MMRRA and IRRVQ to the three algorithms and applying them to the same groups, the results showed that in the range 5-20, Algorithm MMRRA is unique in its outperformance 90% of the time, as shown in Table 4 and Figure 4, in the range 5-50, Algorithm MMRRA is unique in its outperformance in 100% of the times, as shown in Table 5 and Figure 5, and it also outperforms in the range 5-100 by the same previous percentage, as shown in Table 6 and Figure 6. The results also showed that algorithm IRRVQ is unique with the greatest average waiting time in the same proportions of the number of times and for the same ranges as Algorithm MMRRA as shown in tables 4, 5, and 6.

Table 4. Average waiting times for burst time range of 5-20

Burst Time Range 5-20											
Method	Grp1	Grp2	Grp3	Grp4	Grp5	Grp6	Grp7	Grp8	Grp9	Grp10	Average
MMRRA	43.2	60.1	39.1	57.5	40.7	59.5	49.1	59.4	41.3	59.3	50.92
NMARR	56.2	77.1	41.1	65.8	42.8	67.8	64.1	70.2	54.3	77.3	61.67
IRRVQ	60.3	98.5	52.3	92.8	56.9	91.8	77.7	91.2	68.3	96.9	78.67
RRDTQQ	61.2	76.1	48.5	79.6	43.4	73.6	63.1	83.4	59.3	76.3	66.45
IRRPDQ	51.6	76.1	38.8	79.6	43.4	73.6	64.1	83.4	59.3	83.3	65.32

Table 5. Average waiting times for burst time range of 5-50

Burst Time Range 5-50											
Method	Grp1	Grp2	Grp3	Grp4	Grp5	Grp6	Grp7	Grp8	Grp9	Grp10	Average
MMRRA	104.8	89	109.9	102.6	112.2	111.6	101.7	108	84.4	111.5	103.57
NMARR	152.8	111.8	116.2	116.3	169.2	118.2	131.7	114.6	90.1	126.2	124.71
IRRVQ	162.1	144	152.8	165.3	171	173.8	163.5	150.7	122.2	155.6	156.1
RRDTQQ	149.8	110.8	117.4	116.3	164.7	118.8	130.7	115.2	90.1	126.2	124
IRRPDQ	151.3	111.8	116.2	126.9	164.7	117	111	114	99.5	125.6	123.8

Table 6. Average waiting times for burst time range of 5-100

Burst Time Ranges 5-100											
Method	Grp1	Grp2	Grp3	Grp4	Grp5	Grp6	Grp7	Grp8	Grp9	Grp10	Average
MMRRA	248	233.3	169.3	204.4	187.2	204.6	183.4	127.4	134.9	102.3	179.48
NMARR	319	341.3	213.4	216.2	240.3	232.1	214.7	145.6	209.9	120.8	225.33
IRRVQ	399.8	343.7	273.7	255.8	299	333.7	256.6	204.6	210.8	144	272.17
RRDTQQ	348.5	333.8	216.4	220.2	239.3	233.3	196.3	147.4	203.9	121.8	226.09
IRRPDQ	319	332.3	213.4	215.2	240.3	252.3	213.7	148	208.4	122.8	226.54

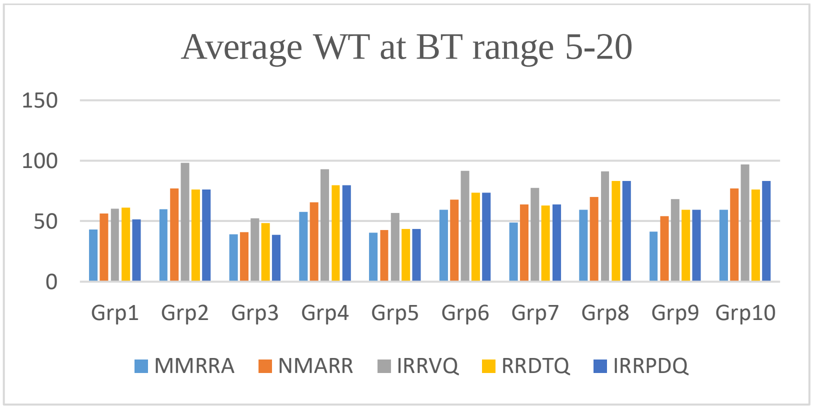


Figure 5. Average waiting times at burst time range of 5-20

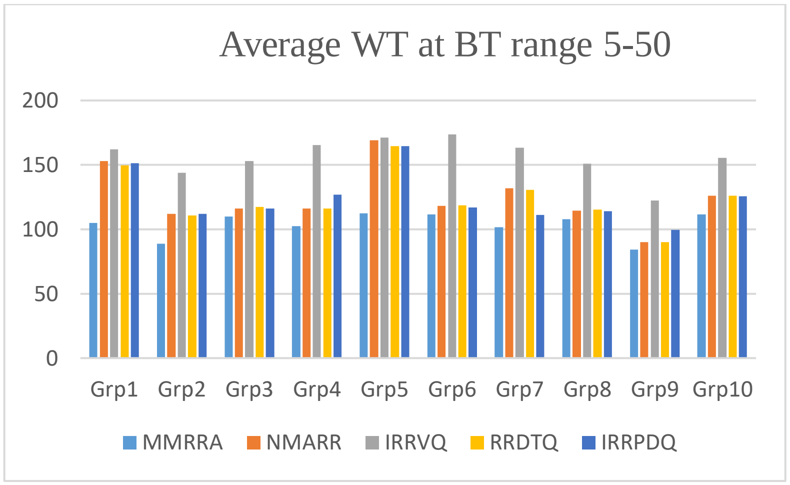


Figure 6. Average waiting times at burst time range of 5-50

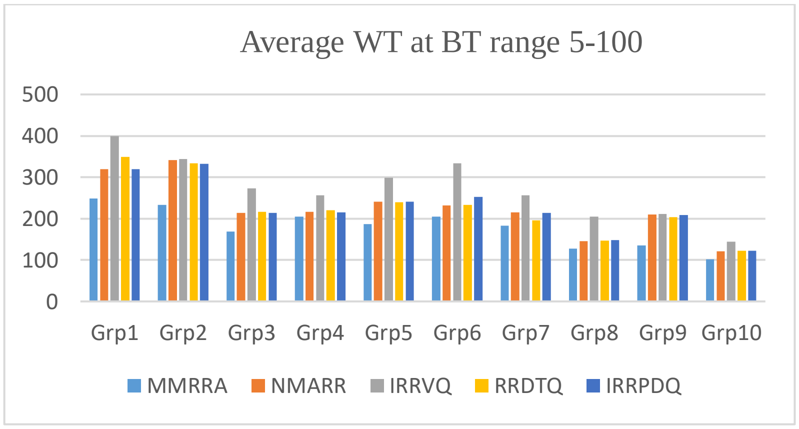


Figure 7. Average waiting times at burst time range of 5-100

Table 7 shows the percentages of the number of individuals and shared outperformance times for each of the three algorithms in each of the ranges.

Table 7. Percentages of times of outperformance in different ranges.

Algorithms	Burst Time Ranges							
	5-20		5-50		5-100		All	
	Solo	Sharing	Solo	Sharing	Solo	Sharing	Solo	Sharing
NMARR	50%	0%	0%	30%	30%	20%	80%	50%
RRDTQQ	20%	10%	20%	30%	30%	0%	70%	40%
IRRPDQ	20%	10%	40%	20%	20%	20%	80%	50%

Table 8 shows the percentages of the number of individuals and shared outperformance times for each of the three algorithms in each of the ranges.

Table 8. Percentages of times of outperformance in different ranges.

Algorithms	Burst Time Ranges							
	5-20		5-50		5-100		All	
	Solo	Sharing	Solo	Sharing	Solo	Sharing	Solo	Sharing
MMRRA	90%	0%	100%	0%	100%	0%	290%	0%
NMARR	0%	0%	0%	0%	0%	0%	0%	0%
IRRVQ	0%	0%	0%	0%	0%	0%	0%	0%
RRDTQQ	0%	0%	0%	0%	0%	0%	0%	0%
IRRPDQ	10%	0%	0%	0%	0%	0%	10%	0%

Conclusion

It is concluded from the results of several algorithms applied to several groups in a certain range, the outperformance can be mutual between those algorithms. Likewise, the outperformance of an algorithm is not absolute, but may be in one range rather than another.

For the three algorithms NMARR, RRDTQQ, and IRRPDQ, we noticed that the percentages of outperformance of the algorithms within a specific criterion are very small. The exchange of outperformance between a group of algorithms and its degree indicates the great convergence in the performance of those algorithms, or at least their convergence within that criterion.

For the five algorithms MMRRA, NMARR, IRRVQ, RRDTQQ, and IRRPDQ, there is a clear indication that MMRRA is the best with the smallest waiting time, and IRRVQ is the worst with the largest waiting time. And the percentages of outperformance are higher than in the case of the three algorithms.

Algorithms MMRRA and IRRVQ have greater relative stability than other algorithms in their behavior with changing ranges of burst times and with changes in their distribution within the same range.

It is also concluded from the results that to measure the performance of an algorithm fairly and accurately:

- The measure should be statistical and not calculated with a specific numerical value.
- It is necessary to indicate the range of values used for the variables.
- Indicating the number of times, the algorithm is applied in one range. The larger the number, the more accurate results.

Indicating the number of processes within one group.

Acknowledgment

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.

Conflict of interest

The author has no conflict of interest.

References

1. P. S. Sharma, S. Kumar, M. S. Gaur, and V. Jain, “A novel intelligent round robin CPU scheduling algorithm,” *Int. J. Inf. Technol.*, vol. 14, no. 3, pp. 1475–1482, 2022, doi: 10.1007/s41870-021-00630-0.
2. 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,” vol. 21, no. 1, pp. 19–26, 2021, doi: 10.22937/IJCSNS.2021.21.1.4.
3. A. Alsheikhy, R. Ammar, and R. Elfouly, “An improved dynamic Round Robin scheduling algorithm based on a variant quantum time,” *2015 11th Int. Comput. Eng. Conf. Today Inf. Soc. What’s Next?, ICENCO 2015*, no. December, pp. 98–104, 2016, doi: 10.1109/ICENCO.2015.7416332.
4. 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.
5. M. Kumar Mishra and F. Rashid, “An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum,” *Int. J. Comput. Sci. Eng. Appl.*, vol. 4, no. 4, pp. 1–8, 2014, doi: 10.5121/ijcsea.2014.4401.
6. 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.
7. 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.
8. H. Mora, S. E. Abdullahi, and S. B. Junaidu, “Modified Median Round Robin Algorithm (MMRRA),” *2017 13th Int. Conf. Electron. Comput. Comput. ICECCO 2017*, vol. 2018-Janua, pp. 1–7, 2018, doi: 10.1109/ICECCO.2017.8333325.
9. S. Mody and S. Mirkar, “Smart Round Robin CPU Scheduling Algorithm for Operating Systems,” *4th Int. Conf. Electr. Electron. Commun. Comput. Technol. Optim. Tech. ICEEECCOT 2019*, no. December 2019, pp. 309–316, 2019, doi: 10.1109/ICEEECCOT46775.2019.9114602.
10. 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.
11. 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.
12. O. S. Samuel, T. O. Omotehinwa, E. Oyekanmi, and E. Olajubu, “An Improved Time Varying Quantum Round Robin CPU Scheduling Algorithm,” no. June 2020.
13. 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.

محاولة وضع معايير لدراسة ومقارنة كفاءة خوارزميات راوند روبن

عبد الناصر يونس أحمد

قسم علوم الحاسوب، كلية التربية للعلوم الصرفة، جامعة الموصل، الموصل، العراق

الخلاصة

مع ظهور الحاجة الى الانظمة التفاعلية برزت الحاجة الماسة الى انظمة تقاسم الوقت. ظهرت خوارزميات راوند روبن لتحقيق تقاسم الوقت.تعتمد درجة اداء انظمة تقاسم الوقت بشكل كبير على طول الشريحة الزمنية في خوارزميات راون روبن. يؤثر طول الشريحة الزمنية في المعايير المستخدمة في اداء الخوارزميات. اقترحت ولا زالت تقترح الخوارزميات لاجل الحصول على القيم الافضل للشريحة الزمنية. ان تبني خوارزمية دون اخرى في نظام ما ولصنف من التطبيقات يتطلب اختيار الخوارزمية الافضل اداء.هذا البحث محاولة لوضع نهج موضوعي للمقارنة الدقيقة بين الخوارزميات. و لاجل الموضوعية في المقارنة، تم اختيار خمس خوارزميات تتشابه في خصائصها العامة ؛ خوارزمية راوند روبن الوسيط المعدلة(MMRRA) ، خوارزمية راوند روبن الوسيط-المعدل الجديدة(NMARR)، خوارزمية جدولة راوند روبن مُحسَّنة مع الكم الزمني المتغير ((IRRVQ، خوارزمية جدولة وحدة المعالجة المركزية راوند روبن المعدلة، خوارزمية جدولة وحدة المعالجة المركزية راوند روبن المعدلة مع الكم الزمني الديناميكي(RRDTQ). اظهرت النتائج ان التفوق لخوارزمية على مجموعة خوارزميات باعتبار معيار معين لا يكون مطلقا،دائميا وثابت القيمة وان اللجوء الى المقاييس الاحصائية هو الافضل في توضيح درجة اداء الخوارزميات.