# Multi-queue CPU Process Prioritization using a Dynamic Quantum Time Algorithm Compared with Varying Time Quantum and Round-Robin Algorithms

Maysoon A. Mohammed
PhD Student
University Malaysia Pahang,
Gambang, Malaysia

Mazlina AbdulMajid
Senior Lecturer
University Malaysia Pahang,
Gambang, Malaysia

Balsam A. Mustafa
Senior Lecturer
University Malaysia Pahang,
Gambang, Malaysia

## ABSTRACT
In Round-Robin Scheduling, the quantum time is static and tasks are scheduled such that no process uses CPU time more than one slice time each cycle. If quantum time is too large, the response time of the processes will not be tolerated in an interactive environment. If quantum the time is too small, unnecessary frequent context switch may occur. Consequently, overheads result in fewer throughputs. In this study, we propose a priority multi queues algorithm with dynamic quantum time. The algorithm uses multi queues with different quantum times for the processes. The quantum times for the processes are depending on the priorities which in turn depending on the burst times of the processes. The proposed algorithm has been compared with varying time quantum algorithm which already exist to improve the original round robin algorithm. With proposed algorithm, the simple Round-Robin algorithm has been improved by about 35%. By controlling quantum time, we experience fewer context switches and shorter waiting and turnaround times, thereby obtaining higher throughput.

## Keywords
Burst Time, Dynamic Quantum Time, Multi queue, Priority, Round Robin.

## 1. INTRODUCTION
Now days multitasking (executing more than one process at a time) and multiplexing (transmitting multiple flows synchronously) are the main processes related with the operating systems. These tasks primarily depends on CPU scheduling algorithm where CPU is one of the important units of operating system. CPU is scheduled by using different algorithms of scheduling which they mean the act of selecting a process from multi running processes to allocate CPU for this process where enable it to access the resources of the system such as processor IO ports, cycles etc. The selected process allocates CPU to a specific period of time which called quantum time which determined by the operating system. The selected process allocates CPU to a specific period of time which called quantum time which determined by the operating system.Now days multitasking (executing more than one process at a time) and multiplexing (transmitting multiple flows synchronously) are the main processes related with the operating systems. These tasks primarily depends on CPU scheduling algorithm where CPU is one of the important units of operating system. CPU is scheduled by using different algorithms of scheduling which they mean the act of selecting a process from multi running processes to allocate CPU for this process where enable it to access the resources of the system such as processor IO ports, cycles etc. The selected process allocates CPU to a specific period of time which called quantum time which determined by the operating system. As researcher [1] previously pointed out that the need for a scheduling algorithm arises from the requirement for fast computer systems to perform multitasking and multiplexing. CPU scheduling is important because it affects resource utilization and other performance parameters [2]. Several CPU scheduling algorithms are available [3], [4], such as First Come First Serve (FCFS), Shortest Job First Scheduling (SJF), Round-Robin (RR) Scheduling, Multilevel queues Scheduling (MQS) and Priority Scheduling (PS). However, due to disadvantages, these algorithms are rarely used in shared time operating systems, except for RR Scheduling [5].

RR is considered the most widely used scheduling algorithm in CPU scheduling [3], [6] also used for flow passing scheduling through a network device [7]. An essential task in operating systems in CPU Scheduling is the process of allocating a specific process for a time slice. Scheduling requires careful attention to ensure fairness and avoid process starvation in the CPU. This allocation is carried out by software known as a scheduler [3], [6].

The scheduler is concerned mainly with the following tasks [8]:

- CPU utilization - to keep the CPU as busy as possible
- Throughput - number of processes that complete their execution per time unit
- Turnaround - total time between submission of a process and its completion
- Waiting time - amount of time a process has been waiting in the ready queue
- Response time - amount of time taken from the time a request was submitted until the production of the first response
- Fairness - equal CPU time allocated to each process

## 2. PERFORMANCE FACTORS
CPU is an essential part in the operating system which is scheduled by many of the scheduling algorithms to keep it busy as much as possible to achieve the perfect utilization of CPU. The processes that need to be processed submit to the system and wait in the ready queue to be selected by the scheduler for the processing. The scheduler is responsible of picking the processes from the ready queue and allocate the CPU if it is idle for that process [2].

The moment that the process joins to the ready queue is called the arrival time. Burst time is the time that the process needs to complete its job inside the CPU. The turnaround time is the time that the process spends in the system from the moment of submission to the moment of completion the processing. Waiting time is the time that the process waits in the ready queue waiting for its turn to be selected by the scheduler for the processing. Therefore, we can conclude that a good scheduling algorithm for real time and time sharing system must possess the following characteristics [9]:

- Minimum context switches
- Maximum CPU utilization
- Maximum throughput
- Minimum turnaround time
- Minimum waiting time

## 3. RELATED WORK

Round Robin Scheduling and multilevel queue scheduling is common in CPU scheduling techniques. The combination between RR and multilevel queue was an interesting subject for many researchers. In [10] developed MLQPTS (Multilevel Queue with Priority & Time Sharing Scheduling) to solve the problem of starvation with real time processes. The processes are scheduled in the queue according to their priority which is defined from the characteristics of the process. Their algorithm can be using multilevel technique because it met the condition of deadline which is the attribute of real time systems. An efficient multi-level round robin multicast scheduling (MLRRMS) algorithm with look ahead (LA) mechanism for N×N input-queued switches has been proposed by [11]. This mechanism can be applied in a parallel fashion with a low time complexity. Related to packet processing networks [12] designed a new cheap multi-resource fair queueing server using O (1) complexity, where the packets have been scheduled in a way similar to elastic RR. Their server is easy to implement, and can be applied in other multi-resource scheduling contexts where jobs must be scheduled as entities. Still in packet networks [13] proposed two downstream multi-channel packet scheduling algorithms designed to support scheduling amongst flows possibly using different numbers of channels. The algorithms provided a low delay for average of packet.

## 4. RR ALGORITHM

RR architecture is a preemptive version of First Come, First Serve scheduling algorithm. The tasks are arranged in the ready queue in first come, first serve manner and the processor executes the task from the ready queue based on time slice. If the time slice ends and the tasks are still executing on the processor, the scheduler will forcibly preempt the executing task and keep it at the end of ready queue. Then, the scheduler will allocate the processor to the next task in the ready queue. The preempted task will make its way to the beginning of the ready list and will be executed by the processor from the point of interruption.

A scheduler requires a time management function to implement the RR architecture and requires a tick timer [14]. The time slice is proportional to the period of clock ticks [8]. The time slice length is a critical issue in real time operating systems. The time slice must not be too small, as it would result in frequent context switches. Moreover, the time slice should be slightly greater than the average task computation time.

RR when implemented in real time operating systems faces two drawbacks, which are high rate of context switch and low throughput. These two problems of RR architecture are interrelated [15].

- Context switch: When the time slice of the task ends and the task is still executing in the processor, the scheduler forcibly preempts the tasks on the processor. The interrupted task is then stored in stacks or registers, and the processor is allocated the next task in the ready queue. This action performed by the scheduler is called "context switch." Context switch leads to wastage of time, memory, and scheduler overhead.
- Larger waiting and response times: In RR architecture, the time the process spends in the ready queue waiting for the processor for task execution is known as "waiting time." The time the process completes its job and exits from the task-set is called "turnaround time." Larger waiting and response times are clearly a drawback in RR architecture, as it leads to degradation of system performance.
- Low throughput: Throughput refers to the number of processes completed per time unit. If RR is implemented in real time operating systems, throughput will be low and results in severe degradation of system performance. If the number of context switches is low, then the throughput will be high. Context switch and throughput are inversely proportional to each other.

## 5. IMPROVED RR WITH VARYING TIME QUNTUM ALGORITHM

The idea of improved Round Robin CPU scheduling algorithm with varying quantum time (IRRVQ) is depending on the combination between Shortest Job First (SJF) and RR with using dynamic quantum time in each round. First, the processes in the ready queue are ordered from lowest to highest burst times. The scheduler allocates the CPU to the first process using RR and assigns its burst time as quantum time for this round. The same procedure will be repeated in each round until all processes finish their execution and ready queue assigns to NULL.

## 6. THE PROPOSED ALGORITHM: MULTIQUEUE DYNAMIC QUANTUM TIME (PMQDQT)

One of the constant challenges for multi queue scheduling is to minimize resource starvation and to ensure fairness amongst the parties utilizing the resources and for real time systems is to build a platform that can meet timeliness requirement of system. RR scheduling algorithm has no priority and fixed quantum time. However, this scheduling algorithm is not suitable for real time operating system (RTOS) because of drawbacks. In other words, the high context switch, high waiting and turnaround times, and low throughput are pitfalls of RR. These disadvantages do not make the optimal choice for RTOS. Priority RR scheduling still has the problem of starvation, where the lowest priority process with fixed quantum time will be starved and preempted by the highest priority process. In multi queue scheduling, the starvation problem has been solved efficiently but this technique is not suitable for real time processes. To overcome this problem, an idea of new algorithm i.e.,

PMQDQT (Priority Multi Queue Dynamic Quantum Time) have been proposed, where the proposed algorithm depends on the existing RR.

## 6.1.Methodology

The proposed algorithm tries to enhance the classic RR by improving the concept of IRRVQ in terms of context switches, average turnaround time and average waiting time with multi queues. In addition, enhancing IRRVQ by prioritizing the processes in the multi ready queues to specify which process from which queue would be chosen by the scheduler for the processing in the CPU. Moreover, changing the quantum time of each queue, rounds and of the processes increases the throughput of the CPU and reduces the waiting time of the processes thus effects as many processes that can be processed by CPU.

## 6.2.The Proposed Algorithm Design

The basic idea of this algorithm considers different priorities depending on the burst times of the processes and different quantum times depending on the priorities [16], [17].

The steps of PMQDQT:

- Allocate multi ready queues for the processes.
- Assign quantum times for the queue such as k.
- Allocate CPU to every process in Round-Robin fashion, according to the given priority and new dynamic quantum time, (for given time quantum k units) only for one time.
- New priorities are assigned according to the CPU bursts of processes; the process with lowest burst time is set with highest priority.
- New quantum times are assigned according to the priorities.
- Calculate new quantum time depending on the existing one by using a simple formula, which is $q = k + n - 1$, where q is the new quantum time for each process, k is the quantum time for each cycle, and n is the priority of the processes in the ready queue.
- Set different quantum times for the processes according to their priorities. The highest priority process will get the largest quantum time, which is q, and the lowest priority process will get the smallest quantum time, which is k.
- The processes in the multi queue that arrive at the same time will be chosen according to their lower burst time.
- Each process gets the control of the CPU until they finished their execution.
- Apply the original RR, improved RR and proposed algorithm with the priorities and new different quantum times.
- Calculate context switches, average turnaround time and average waiting time.

By changing the quantum time for the cycles and processes, we guarantee that one or more processes complete their jobs every cycle. Also, we could improve the existing RR algorithm by reducing context switches and lessening turnaround and waiting times. Hence, throughput will increase. The next sections present case studies to show the differences between PMQDQT, IRRVQ and RR Algorithms.

# 7. EXPERMINTAL SIMULATION

## 7.1 Assumptions

The assumptions that we followed in the case studies are: The Quantum time has been taken in milliseconds, the CPU bound is active that mean all processes are in CPU bound not in I/O bound. For IRRVQ all processes with the same priorities while in our algorithm different priorities used for all processes. For experimental purposes, the burst times and arrival times of all processes are known and chosen by the researchers. The context switches in IRRVQ are considered zero while in PDQT are computed. The overhead of arranging the ready queue processes in ascending order has been considered zero in IRRVQ [18] as well in PMQDQT.

## 7.2 Case Study

Two queues Q1 and Q2 with six and five processes in Q1 and Q2 respectively have been defined with CPU burst times, different arrival times, and their priorities. These processes are scheduled in RR, IRRVQ techniques as well as according to the PMQDQT algorithm. The context switches, average turnaround time, and average waiting time are calculated, and the results are compared. To accomplish this task, we implemented the algorithm in JAVA programming language and conducted several experiments. However, only one experiment is discussed here for dynamic quantum time process, and we assure that the analysis remain the same for the other experiments.

We consider Q1 with six processes (A1, A2, A3, A4, A5, and A6) with quantum time 4 millisecond and Q2 with five processes (B1, B2, B3, B4, and B5) with quantum time 4 millisecond. Different arrival times, and burst times as shown in Table1.

The equations used to calculate average turnaround and average waiting time are:

$$\text{Average turnaround time} = \sum_{k=1}^{n} T/n \quad (1)$$

$$\text{Average waiting time} = \sum_{k=1}^{n} B/n \quad (2)$$

, where n = number of processes, T = completion time – arrival time; B = turnaround time – burst time

The processes with arrival and burst times are shown in table 1. Tables 2, 3 and 4 show the output of using algorithms RR, IRRVQ and PMQDQT respectively.

**Table1: The inputs for the processes of case study**

| Tasks of Q1 | AT | BT | QT | Tasks of Q2 | AT | BT | QT |
|---|---|---|---|---|---|---|---|
| A1 | 0 | 4 | 4 | B1 | 0 | 3 | 4 |
| A2 | 5 | 8 | 4 | B2 | 7 | 5 | 4 |
| A3 | 10 | 12 | 4 | B3 | 10 | 10 | 4 |
| A4 | 15 | 10 | 4 | B4 | 25 | 7 | 4 |
| A5 | 20 | 6 | 4 | B5 | 30 | 13 | 4 |
| A6 | 25 | 15 | 4 | | | | |

**Table2: Output processes of RR**

| Round | QT of Round for Q1 | QT of Round of Q2 | Pi, QTi | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 4 | A1=4 | B1=3 | A2=4 | B2=4 | A3=4 | B3=4 | A4=4 | A5=4 | A6=4 | B4=4 | B5=4 |
| 2 | 4 | 4 | A2=4 | B2=1 | A3=4 | B3=4 | A4=4 | A5=2 | A6=4 | B4=3 | B5=4 | | |
| 3 | 4 | 4 | A3=4 | B3=2 | A4=2 | A6=4 | B5=4 | | | | | | |
| 4 | 4 | 4 | A6=3 | B5=1 | | | | | | | | | |

**Table3: Output processes of IRRVQ**

| Round | QT of Round for Q1 | QT of Round of Q2 | Pi, QTi | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 4 | A1=4 | B1=3 | A2=4 | B2=4 | A3=4 | B3=4 | A4=4 | A5=4 | A6=4 | B4=4 | B5=4 |
| 2 | 4 | 4 | A2=4 | B2=1 | A3=4 | B3=4 | A4=4 | A5=2 | A6=4 | B4=3 | B5=4 | | |
| 3 | 4 | 4 | A3=4 | B3=2 | A4=2 | A6=4 | B5=4 | | | | | | |
| 4 | 3 | 3 | A6=3 | B5=1 | | | | | | | | | |

**Table4: Output processes of PMQDQT**

| Round | QT of Round for Q1 | QT of Round of Q2 | Pi, QTi | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 4 | B1=3 | A1=4 | A2=8 | B2=5 | B3=7 | A3=6 | A4=8 | A5=6 | B4=7 | A6=4 | B5=5 |
| 2 | 4 | 4 | B3=3 | A3=6 | A4=2 | A6=4 | B5=5 | | | | | | |
| 3 | 4 | 4 | A6=4 | B5=3 | | | | | | | | | |
| 4 | 4 | 4 | A6=3 | | | | | | | | | | |

Figures 1, 2 and 3 show Gantt charts of the three algorithms RR, IRRVQ and PDQT respectively.

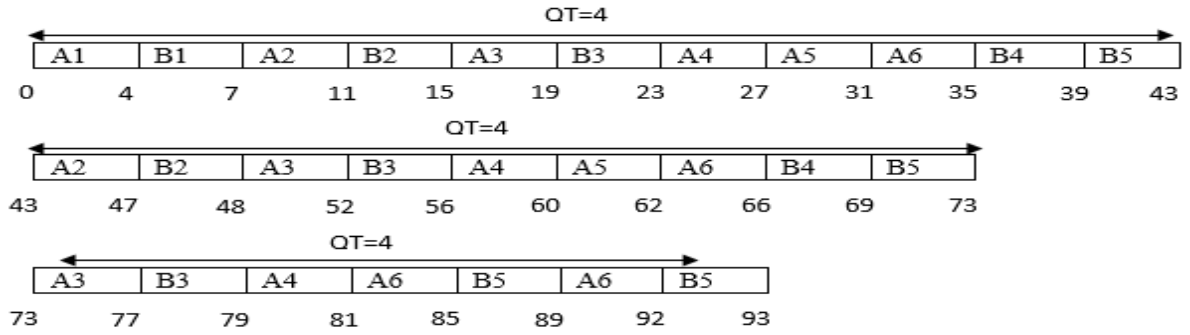


**Fig. 1: Gantt chart of RR**

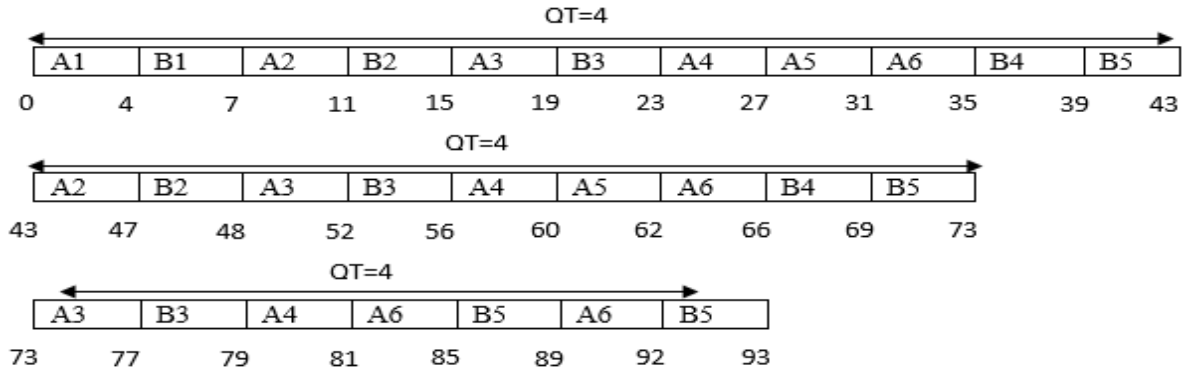


**Fig. 2: Gantt chart of IRRVQ**

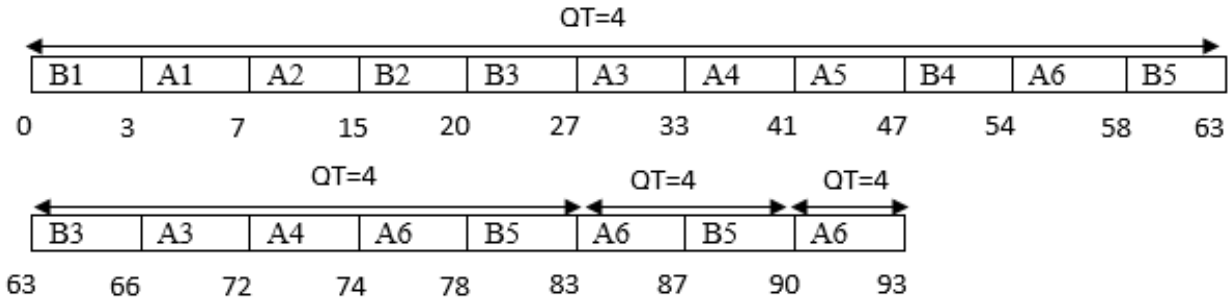


**Fig. 3: Gantt chart of PMQDQT**

Figures above show the staging performance of the three algorithms, where we note the performance of RR remain exactly the same when applying IRRVQ in this case study. On the other hand, the proposed algorithm PMQDQT improved and raised the level of performance when reducing the context switches according to the both algorithms, the original RR and IRRVQ.

## 8. RESULTS AND COMPARISON

The results that conducted after applying the three algorithms RR, IRRVQ and PMQDQT are shown in table 5.

**Table5: Results obtained from the three algorithms**

| Algorithm | Average TAT | Average WT | CS |
|---|---|---|---|
| RR | 47 | 38 | 26 |
| IRRVQ | 45 | 38 | 26 |
| PMQDQT | 37 | 28 | 18 |

From the results above, it is obvious that the proposed algorithm PMQDQT conducted results with context switches, average turnaround and average waiting time much better than RR and IRRVQ. Figures 4, 5 and 6 illustrate the comparison of performance of RR, IRRVQ and PMQDQT algorithms for 5 different quantum times in terms of the three factors, context switches, average turnaround and average waiting times, respectively.

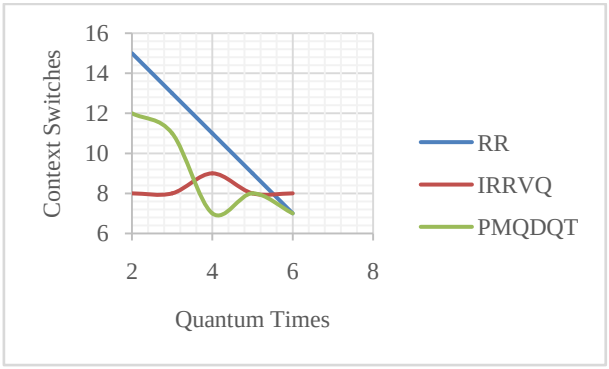


**Fig. 4: Performance of RR, IRRVQ and PMQDQT in term of Context Switches**

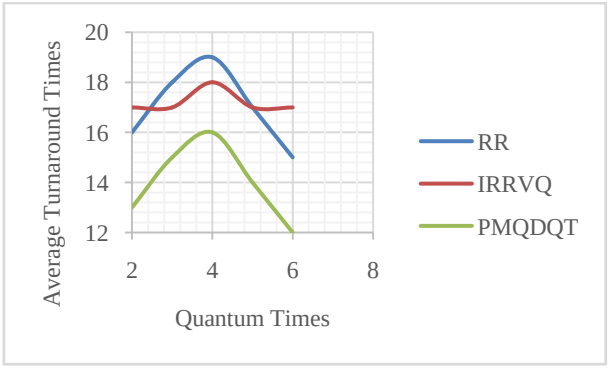


**Fig. 5: Performance of RR, IRRVQ and PMQDQT in term of average Turnaround Times**

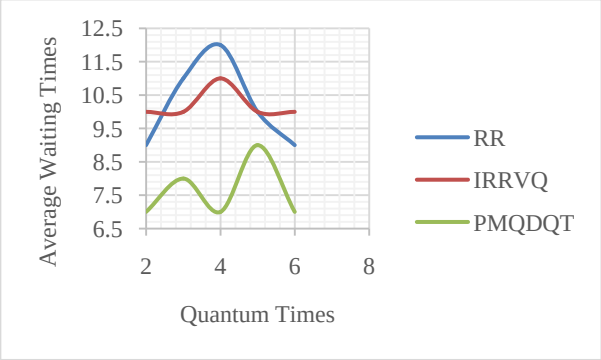


**Fig. 5: Performance of RR, IRRVQ and PMQDQT in term of average Waiting Times**

After applying the algorithms with different quantum times we conduct a conclusion with the improved algorithm IRRVQ, which is, for some quantum times IRRVQ did not accomplish results better than RR, on contrast, PMQDQT in every experiment for any quantum time achieved the proper results over than RR and IRRVQ. So, PMQDQT solved the problems that can face us with IRRVQ. On the other hand, a limitation faces us with PMQDQT, it is the large values of burst times with large number of context switches. But this problem do not affect too much because the processes in worst cases do not need a large burst time to complete their job.

Honestly and for scientific secretariat, this work may be relative to the work of [16] and [17], but the difference of their work is the novelty of the new formula that used in the proposed algorithm and the dependence of the dynamic quantum time of the processes upon priorities.

## 9. CONCLUSIONS AND THE FUTURE WORKS

We have successfully compared three algorithms, namely, simple RR, Improved algorithm IRRVQ and the proposed algorithm PMQDQT for multi queues with priorities according to process's burst time and dynamic quantum time according to the priority of the process. Results indicated that PMQDQT is more efficient because this proposed algorithm has fewer context switches and shorter average turnaround and waiting times compared to simple RR and IRRVQ. Moreover, the results reduced operating system overhead and increased throughput. PMQDQT lessened the problem of starvation as the processes with highest priorities are assigned to the lowest CPU burst time with largest quantum time and are executed before lower priority processes.

After experience many quantum times with the three algorithms, some important points have been conducted and listed below:

1. With IRRVQ, if the quantum time is large than the burst time of the first process in the ready queue, the results that conducted with IRRVQ stay static, e.g. No real improvement of the performance with this algorithm.
2. The performance of IRRVQ is weak, sometimes give the same results with RR, if the processes arrive in different arrival times.
3. IRRVQ gives much better performance over RR with zero arrival times for the processes.
4. PMQDQT, gives better results and performance with different arrival times and different quantum times.
5. The advantage of PMQDQT algorithm is, high performance with the large number of processes which will be the next improvement of the algorithm to compare with other techniques, however, there is a limitation faces us is the low performance with the large burst times with high quantum time.

For the future works, the performance of time-sharing systems can be improved with the proposed algorithm, and can be modified to enhance the performance of real time system. Moreover, the idea of applying the proposed algorithm in real environment, operating system such as Linux, is under study in order to achieve the objective of improving RR algorithm.

## 10. REFERENCES

[1] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*: Springer, 2011.

[2] T. F. Hasan, "CPU SCHEDULING VISUALIZATION."

[3] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating system concepts* vol. 8: Wiley, 2013.

[4] E. Oyetunji and A. Oluleye, "Performance Assessment of Some CPU Scheduling Algorithms," *Research Journal of Information and Technology*, vol. 1, pp. 22-26, 2009.

[5] F. Cerqueira and B. Brandenburg, "A comparison of scheduling latency in linux, PREEMPT-RT, and LITMUSRT," in *Proceedings of the 9th Annual Workshop on Operating Systems Platforms for Embedded Real-Time applications*, 2013, pp. 19-29.

[6] L. Yang, J. M. Schopf, and I. Foster, "Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environments," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, 2003, p. 31.

[7] W. Tong and J. Zhao, "Quantum varying deficit round robin scheduling over priority queues," in *Computational Intelligence and Security, 2007 International Conference on*, 2007, pp. 252-256.

[8] M.-X. Chen and S.-H. Liu, "Hierarchical Deficit Round-Robin Packet Scheduling Algorithm," in *Advances in Intelligent Systems and Applications-Volume 1*, ed: Springer, 2013, pp. 419-427.

[9] A. Singh, P. Goyal, and S. Batra, "An Optimized Round Robin Scheduling Algorithm for CPU Scheduling," *IJCSE) International Journal on Computer Science and Engineering*, vol. 2, pp. 2383-2385, 2010.

[10] I. Sattar, M. Shahid, and N. Yasir, "Multi-Level Queue with Priority and Time Sharing for Real Time Scheduling."

[11] H. Yu, S. Ruepp, and M. S. Berger, "Multi-level round-robin multicast scheduling with look-ahead mechanism," in *Communications (ICC), 2011 IEEE International Conference on*, 2011, pp. 1-5.

[12] W. Wang, B. Li, and B. Liang, "Multi-Resource Round Robin: A low complexity packet scheduler with Dominant Resource Fairness," in *ICNP*, 2013, pp. 1-10.

[13] D. Nikolova and C. Blondia, "Bonded deficit round robin scheduling for multi-channel networks," *Computer Networks*, vol. 55, pp. 3503-3516, 2011.

[14] N. Goel and R. Garg, "An Optimum Multilevel Dynamic Round Robin Scheduling Algorithm," *arXiv preprint arXiv:1307.4167*, 2013.

[15] R. S. Kiran, P. V. Babu, and B. M. Krishna, "Optimizing CPU scheduling for real time applications using mean-difference round robin (MDRR) algorithm," in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I*, 2014, pp. 713-721.

[16] I. S. Rajput and D. Gupta, "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems," *International Journal of Innovations in Engineering and Technology*, 2012.

[17] R. Mohanty, H. Behera, K. Patwari, M. Dash, and M. L. Prasanna, "Priority based dynamic round robin (PBDRR) algorithm with intelligent time slice for soft real time systems," *arXiv preprint arXiv:1105.1736*, 2011.

[18] M. K. Mishra and F. Rashid, "AN IMPROVED ROUND ROBIN CPU SCHEDULING ALGORITHM WITH VARYING TIME QUANTUM," *International Journal of Computer Science, Engineering & Applications*, vol. 4, 2014.