An Effective Dynamic Quantum Round Robin (EDQRR) CPU Scheduling Algorithm

Kankana Datta  
Haldia Institute of Technology  
ICARE Complex  
Haldia, WB, India

Manasi Jana  
Haldia Institute of Technology  
ICARE Complex  
Haldia, WB, India

Arpita Mazumdar  
Haldia Institute of Technology  
ICARE Complex  
Haldia, WB, India

ABSTRACT
Processor scheduling is one of the primary task of time sharing operating system. The main goal behind CPU scheduling algorithm is to identify the process and assign it to CPU that will give the best possible system performance. In this report, a dynamic approach is presented for Round Robin CPU scheduling. Here a modified concept of dynamic quantum for each round is designed to achieve lesser average waiting time, lesser average turnaround time and lesser number of context switching as an improved feature to conventional Round Robin scheme and also a few number of existing schemes. The comparative study of processing performance clearly shows that the proposed scheme gives approximate [15 - 20]% better result than few related recent works.

Keywords
dynamic time quantum, average turnaround time, average waiting time, context switching

1. INTRODUCTION
In time sharing computing system, the CPU time is a valuable resource. In this environment, several processes wait in the ready queue of main memory. CPU utilisation may be optimised by allocating CPU time among the waiting processes and keep the CPU busy always. Now, selection of the processes for CPU attention should be done in such a manner so that performance will be optimised. Scheduling decision try to reduce the following: turnaround time, response time and average waiting time for processes and the number of context switches. Numerous scheduling algorithms have been proposed [1][2][6][9]. In this paper, a dynamic quantum based Round Robin algorithm is proposed where the average waiting time, average turnaround time and number of context switching are reduced significantly and increase the throughput in comparison to [1][2].

1.1 Performance Criteria
To monitor the processor performance [3] the following performance parameters are to be measured. The various criteria are listed below:

1. CPU Utilization: It is the percentage of time, the CPU executes a process.
2. Throughput: The number of processes complete their execution per time unit.
3. Turnaround time: The amount of time to execute a particular process.
4. Waiting time: The amount of time a process has been waiting in the ready queue.
5. Response time: It is the difference between time of submission of process and the time the first response occurs.

1.2 CPU Scheduling Algorithm
Scheduling algorithm differ in the manner in which the CPU selects a process in the ready queue for execution. Few fundamental scheduling algorithms are discussed below:

1. FCFS (First Come First Serve):
FCFS is the simplest scheduling algorithm. In this algorithm, the process that request CPU first is allocated the CPU first. Here the ready queue is organized as a FIFO queue. When a process enters into the ready queue, the PCB (Process Control Block) of a process is linked to the tail of ready queue. The algorithm dispatches processes from the head of the ready queue for execution. After completion the process is deleted from the system. The next process is then dispatched from the ready queue.

2. SJF (Shortest Job First):
In this scheduling scheme, the process with the smallest CPU burst time is allocated to the CPU first. If multiple processes having the same CPU burst exists in the ready queue, then CPU is allocated to the process on FCFS basis. After completion the process is deleted from the system. The SJF algorithm can be preemptive or non-preemptive in nature. The processes here suffer from starvation. This problem can be solved by aging technique.

3. Priority Based Scheduling:
In this CPU scheduling, priority is associated with each process and on the basis of priority the CPU is allocated to the processes. The process with the highest priority is allocated to the CPU first. If multiple processes having the same priority exist in the ready queue, the control of CPU is allocated to these processes on FCFS basis. The priority scheduling can be preemptive or non-preemptive in nature. The processes here suffer from starvation. This problem can be solved by aging technique.
4. Round Robin Scheduling:
Round Robin Scheduling is used in time-sharing systems. In this algorithm, the ready queue is maintained as a FIFO queue. Here a small unit of time, called a time quantum or time slice is defined by the system. Here the first process in the ready queue is dispatched for execution and is preempted on expiry of one time quantum. Then the preempted process is inserted at the tail of the ready queue. When a process has completed its task, it is deleted from the system. The next process is dispatched from the ready queue. The performance of the RR algorithm is very much dependent on the length of the time slice. If the duration of the time slice is indefinitely large then RR algorithm is the same as FCFS algorithm. If the time slice is too small, then the performance of the algorithm deteriorates because of the effect of frequent context switching.

2. RELATED WORKS
The traditional RR scheduling has a disadvantage that it uses static time quantum. To eliminate this disadvantage various modifications on RR scheduling have been done by several authors introducing dynamic time quantum. Several related works are discussed as follows:-
Raja Ram Jaiswal et al. [1] proposed a new approach on dynamic time quantum which is repeatedly changed to the immediate greater value of the previous time quantum. The result of this paper shows a significant improvement in terms of reduction in waiting time, turnaround time and context switches as compared to RR scheduling.

A new approach IRRVQ [2] algorithm uses dynamic time quantum which is repeatedly adjusted to the minimum value of retaining burst time. This approach gives a significant performance improvement compared to RR. Nayana Kundargi [5] has proposed NK algorithm which calculate the average of all remaining burst time to assign to dynamic time quantum. EDRR [6] algorithm uses dynamic time quantum which is calculated by the formula cell (sqrt((mean*highest burst time)+(median* lowest burst time))) giving better result than RR. In [7] the author has proposed an algorithm by introducing a concept of assigning different time quantum by calculating some metrics like RRB, WR, PF etc. In [8] ARR has been proposed by calculating smart time slice depending on number of processes. Abbas Noon [9] developed an algorithm which calculates mean average to improve the scheduling algorithm performance. Saroj Hiranwal [10] proposed the same concept as RR calculating smart time slice using shortest burst time approach. In [11] AMRR has been developed by setting dynamic time quantum to (AVG+MAX BT)/2 to improve RR scheduling.

3. PROPOSED WORK
The Effective Dynamic Quantum Round Robin(EDQRR) CPU scheduling algorithm is based on RR and SJF scheduling. This algorithm eliminates the drawback of round robin algorithm in which processes are scheduled in FCFS manner. The new proposed algorithm uses the dynamic time quantum to minimize the average turnaround time, the average waiting time and context switches.

First, processes in the ready queue are arranged in ascending order based on CPU burst time. Then quantum time is initialized to the minimum burst time of all processes in the ready queue and allocate the CPU to the first process for 1 time quantum. After completion, if remaining CPU burst time of the currently running process is less than or equal to the time quantum, the CPU is again allocated to the currently running process for remaining CPU burst time. Otherwise, it is inserted at the tail of the ready queue. The scheduler then allocate CPU to the next process in the ready queue.

3.1 Flow Chart

![Flow Chart of the proposed Algorithm(EDQRR)](image)

3.2 Algorithm
Assuming all the processes arrive at same time.

**Step 1:** Start

**Step 2:** Initialization

- np // Number of processes in the ready queue
- bt[np] // Burst time of the processes
- rbt[np] // Remaining Burst Time of the Processes
- qt // Quantum Time
- cs // Context Switch
- awt // Average Waiting Time
- att // Average Turn Around Time

**Step 3:** According to the bt , arrange the process in ascending order in the Ready Queue

**Step 4:** if(one or more processes has equal or same bt )

Allocate the CPU to the process according to FCFS scheme

End if

**Step 5:** while(Ready Queue!=NULL)

**[5.1]** Assign qt=minimum(rbt[np])

**[5.2]** Execute all the processes with present qt

if(qt>=bt[i])

Execute the processes with present qt and delete it from Ready Queue as it is terminated

else

Execute the processes with present qt if(rbt[i]=qt)

Execute the same process again
else 
Insert the process at the tail of the Ready Queue 
and Go to Step 5 
end if 
end if 
end while 

Step 6: Calculate CS, AWT, ATT 

Step 7: Stop. 

3.3 Illustrative Example 

It is considered that the ready queue with 4 processes P1,P2,P3 
and P4 arrive at the same time with burst time 10,12,22,30 ms 
respectively as given in Table 1. 

<table>
<thead>
<tr>
<th>Process name</th>
<th>CPU burst time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>10</td>
</tr>
<tr>
<td>P2</td>
<td>12</td>
</tr>
<tr>
<td>P3</td>
<td>22</td>
</tr>
<tr>
<td>P4</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 1. Input table 

3.4 Result analysis 

For the above mentioned data set let the time quantum of simple RR is 10 ms. The Gantt chart of simple RR is given in Fig 2. 

Waiting time of each process is calculated as following:- 
P1= 0 ms 
P2= 30 ms 
P3= 42 ms 
P4= 44 ms 
Average Waiting Time(AWT)= (0+30+42+44)/4= 29 ms. 

Turn Around time of each process is as following:- 
P1= 10 ms 
P2= 42 ms 
P3= 56 ms 
P4= 74 ms 
Average Turn Around Time(ATT)= (10+42+56+74)/4= 47.5 ms. 

Context Switch(CS)=8 

The Gantt chart of IRRVQ is given in Fig 3. 

Waiting time of each process is calculated as following:- 
P1= 0 ms 
P2= 30 ms 
P3= 34 ms 
P4= 44 ms 
Average Waiting Time(AWT)= (0+30+34+44)/4= 27 ms. 

Turn Around time of each process is as following:- 
P1= 10 ms 
P2= 42 ms 
P3= 56 ms 
P4= 74 ms 
Average Turn Around Time(ATT)= (10+42+56+74)/4= 45.5 ms. 

Context Switch(CS)=8 

The Gantt chart of IARR is given in Fig 4. 

Waiting time of each process is calculated as following:- 
P1= 10 ms 
P2= 42 ms 
P3= 56 ms 
P4= 74 ms 
Average Waiting Time(AWT)= (10+42+56+74)/4= 45 ms. 

Context Switch(CS)=6 

The Gantt chart of Proposed Algorithm(EDQRR) is given in Fig 5. 

Waiting time of each process is calculated as following:- 
P1= 0 ms 
P2= 10 ms 
P3= 42 ms 
P4= 44 ms 
Average Waiting Time(AWT)= (0+10+42+44)/4= 26.5 ms. 

Turn Around time of each process is as following:- 
P1= 10 ms 
P2= 44 ms 
P3= 56 ms 
P4= 74 ms 
Average Turn Around Time(ATT)= (10+42+54+74)/4= 45 ms. 

Context Switch(CS)=6 

The Gantt chart of Proposed Algorithm(EDQRR) is given in Fig 5.
Fig. 5. Gantt Chart of the proposed Algorithm (EDQRR)

P3= 32 ms  
P4= 44 ms  
Average Waiting Time (AWT) = (0+10+32+44)/4 = 21.5 ms.

Turn Around time of each process is as following:-  
P1= 10 ms  
P2= 22 ms  
P3= 54 ms  
P4= 74 ms  
Average Turn Around Time (ATT) = (10+22+54+74)/4 = 40 ms.

Context Switch (CS) = 5

3.5 Comparison

The proposed EDQRR algorithm has been simulated along with the existing RR, IRRVQ, and IARR for performance comparison using Dev-C++. The following metrics viz. average waiting time (AWT), average turnaround time (ATT), number of context switches (CS) are measured for different time quantum (TQ). The comparison result of RR, IRRVQ, IARR and our proposed EDQRR algorithm are shown in Table 2 and Fig. 6.

Table 2. Comparison between RR, IARR, IRRVQ and proposed EDQRR

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>TQ</th>
<th>AWT</th>
<th>ATT</th>
<th>No. of CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>10</td>
<td>29</td>
<td>47.5</td>
<td>8</td>
</tr>
<tr>
<td>IRRVQ</td>
<td>10,2,10,8</td>
<td>27</td>
<td>45.5</td>
<td>8</td>
</tr>
<tr>
<td>IARR</td>
<td>10,12,8</td>
<td>26.5</td>
<td>45</td>
<td>6</td>
</tr>
<tr>
<td>Proposed(EDQRR)</td>
<td>10,12,8</td>
<td>21.5</td>
<td>40</td>
<td>5</td>
</tr>
</tbody>
</table>

3.6 Conclusion

From the comparison table it can be inferred that the proposed EDQRR algorithm produces better performance than conventional fixed quantum Round Robin algorithm, IRRVQ and IARR algorithm with dynamic time quantum in terms of average waiting time, average turnaround time and number of context switching. This approach can be further enhanced considering different arrival time of processes. In future, this method would be modified using the other methods of artificial intelligence such as Fuzzy Logic, Neural Network by the authors. It may improve the performance of the operating system by using more advanced CPU scheduling algorithm.

4. REFERENCES