Improved Round Robin CPU Scheduling Algorithm

Round Robin, Shortest Job First and Priority algorithm coupled to increase throughput and decrease waiting time and turnaround time

Harshal Bharatkumar Parekh, Sheetal Chaudhari
harshalparekh@outlook.com, sheetal_chaudhari@spit.ac.in
Sardar Patel Institute of Technology, Andheri, Mumbai

Abstract—**In multitasking operating system, processes don't run simultaneously, but switch very expeditiously. Thus, all the processes share the CPU time. It is the job of the scheduler to select a process from the ready queue and place it into the memory based a particular strategy known as Scheduling Algorithm. There exist many Scheduling Algorithms such as First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Priority Scheduling, Multilevel Queue Scheduling (MLQ). This paper proposes a new strategy consolidating three of the existing scheduling algorithms to create a new strategy that reduces the time a process spends in waiting state. The proposed algorithm also reduces the number of context switches to provide a fair, efficient and methodical scheduling algorithm. The algorithm is an extended version of Round Robin Algorithm where each of the processes is given a priority level (low, medium or high) and based on the priority level, the Time Quantum for that process is decided and executed. The throughput of the algorithm is increased by executing processes with smaller burst time or smaller remaining burst time by allocating the process to the CPU as soon as it is ready but not pre-empting the running process.**

Keywords—RR, SJF, Priority, Waiting Time, Turn Around Time, Context Switch, CPU Scheduling, Time Quantum.

I. INTRODUCTION

According to the Round Robin [1] algorithm, it gives each of the process a fair time slice or time quantum of the CPU. The time slice should not be too short as it would increase the number of context switches and should not be long as it can would cause starvation of other processes. In Shortest Job First algorithm, the process with shortest CPU burst in the ready queue, it allotted the CPU first. Although the algorithm has the best throughput, it causes the processes with longer burst time to starve. The Priority [1] based algorithm, the high priority processes are allotted to CPU before the low priority processes, if two or more processes have the same priority, then CPU is allotted on First Come First Serve (FCFS) basis. Such prioritizing of the processes leads to starvation of lower priority processes. The proposed algorithm 'Improved Round Robin' uses the Round Robin algorithm integrated with a consolidation of Shortest Job First algorithm and Priority based algorithm. The Scheduler [2] uses one of such algorithms and manages the CPU time for each process. There are 3 types of Schedulers: Long term, Short term, and Medium term. The Long term scheduler is also known as Job Scheduler as it takes a process from the new state into the ready state. The Short term scheduler is also known as the CPU scheduler as it takes a process from the ready state into the running state and vice versa. The Medium term scheduler swaps a process in and out of memory to avoid starvation of other processes in the Job queue. Context Switch is a process of storing the state and data values of a process of a thread so that the execution can be restored later. It is an overhead for the Operating System as all the register values and data has to be stored in a Process Control Block (PCB) [2] and maintain all the PCBs. Thus, the number of Context Switches should be reduced.

II. PROPOSED ALGORITHM

The algorithm combines three of the existing algorithms into one. All the processes in the job queue have a priority. The priority ranges from low, medium to high. According to the Round Robin algorithm, a fair time slice is given to all the processes. Thus, the processes with low priority have smaller time slice than the medium priority processes and the medium priority processes have a smaller time slice than the high priority processes. Example, if the time slice is 500ms, then the low priority processes should get 400ms of CPU and the high priority processes should get 600ms of CPU time. Now, if a process with very low burst time arrives into the ready queue and has high priority, then it should perform its execution soon after the execution of the ongoing process, thus increasing the throughput. Example, if a process with Burst time 200ms arrives in the ready queue at n^{th} interval, it should perform its execution soon after the execution of the ongoing process. If the execution time that remains after the process was allotted CPU once is very less, then it should finish its entire execution in that time slice. Example, if a process of medium priority has 600ms of execution left, it performs the execution for 500ms, now the remaining execution time is 100ms should also be executed, thus reducing the number of context switches.

The authors [3] have mainly focused on dynamic time quantum but it causes context switches only for a fraction of remaining burst time which is an overhead for the operating system. Improved Round Robin algorithm uses the priority of the processes to assign the time quantum of each priority statically. The algorithm also reduces the number of context switches by completing the execution when the remaining burst time is very less.

A. Selection of the Algorithm

Different kinds of Operating Systems have different kinds of scheduling algorithm based on the purpose of application of the Operating System. Selection of a scheduling algorithm depends on various selection factors such as efficiency (the wellness in execution of processes), fairness (fair CPU allocation), number of context switches (number of saving and retrieving the process information from the PCB), starvation (time spent in ready queue before getting the CPU for the first time), average waiting time (amount of time a process waits in its lifetime), average turnaround time (the amount of time spent in ready queue and amount of execution time) and response time (time taken by process to attain CPU) [4]. Based on the type of applications to be used in the system, some of these factors are taken into consideration to select the appropriate scheduling algorithm. Improved Round Robin algorithm serves the purposes by reducing the number of context switches, giving fair time slice of the CPU to all the processes, avoiding starvation of processes, reducing the time a process spends in waiting state. The algorithm differentiates between the high priority processes and low priority processes which is an advantage in prioritizing the system processes as high priority processes should be given higher weight-age than the low priority processes.

III. ALGORITHM

The following variables must be defined before executing the algorithm. Each of the process should have its own Burst Time (BT), Arrival Time (AT), Waiting Time (WT), Turnaround Time (TAT), Priority (P) and Process ID. The algorithm has static variable Time Quantum and general variables Average Waiting Time (AWT) and Average Turnaround Time (ATT).

Algorithm:

1. Input number of processes and define essential variables.

2. Input BT, AT, P for each Process.

3. Initialize the flag of each Process in the Job Queue(JQ) to FALSE.

4. Input Time Quantum (TQ).

5. Assign appropriate TQ for each process based on the priority i.e. Medium Priority = TQ,
   Low Priority = (TQ – (20%*TQ)) and
   High Priority = (TQ + (20%*TQ)).

6. If there is any process with very low BT in the Ready Queue (RQ), then assign that process to the CPU and finish its execution. Mark its flag to TRUE and calculate its waiting time and turnaround time and remove it from the RQ.

7. Else allot the CPU in Round Robin to the next process in the RQ.

8. If (BT <= TQ) then execute the entire process for its remaining burst time. Mark its flag to TRUE and calculate its waiting time and turnaround time and remove it from the RQ (the TQ is different for different priority processes).

9. if ((BT > TQ) && BT <= (TQ + (30%*TQ)) && if (P == HIGH) then execute the entire process for its remaining burst time. Mark its flag to TRUE and calculate its waiting time and turnaround time and remove it from the RQ.

10. if ((BT > TQ) && BT <= (TQ + (20%TQ)) && P == (MEDIUM or LOW)) then execute the entire process for its remaining burst time. Mark its flag to TRUE and calculate its waiting time and turnaround time and remove it from the RQ.

11. Else execute the process for its assigned TQ and switch to next process in the RQ until RQ is empty.

12. Calculate the AWT, ATT and Number of Context Switches and display the result.

IV. EXAMPLES

A. Example 1:

TABLE I.

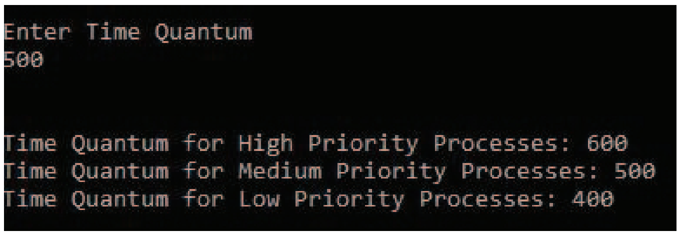
Process	Burst Time	Arrival Time	Priority
1	550	0	3
2	800	200	1
3	200	100	3
4	2600	400	2
5	1600	0	2

Fig 1: Burst time, arrival time and priorities of processes

<u>By the Round Robin algorithm:</u> (Time Quantum=500)
Average Waiting Time = 2090.0 units
Average Turnaround Time = 3240.0
units Number of Context Switches = 13

<u>By Improved Round Robin algorithm:</u> (Time Quantum= 500)

Number of Context Switches = 10

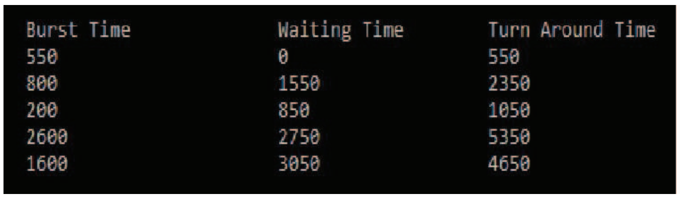


```
Enter Time Quantum
500


Time Quantum for High Priority Processes: 600
Time Quantum for Medium Priority Processes: 500
Time Quantum for Low Priority Processes: 400
```

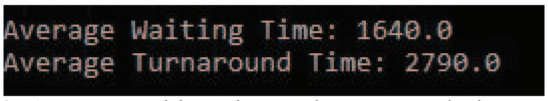
Fig 1.1: Time Quantum for each Priority

Results:



```
Burst Time          Waiting Time        Turn Around Time
550                 0                   550
800                 1550                2350
200                 850                 1050
2600                2750                5350
1600                3050                4650
```

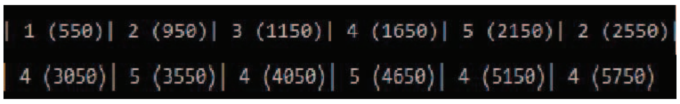
Fig 1.2: Individual Waiting time and Turnaround Time



```
Average Waiting Time: 1640.0
Average Turnaround Time: 2790.0
```

Fig 1.3: Average Waiting Time and Turnaround Time

Gantt Chart:



```
| 1 (550)| 2 (950)| 3 (1150)| 4 (1650)| 5 (2150)| 2 (2550)|
| 4 (3050)| 5 (3550)| 4 (4050)| 5 (4650)| 4 (5150)| 4 (5750)
```

The Round Robin algorithm lags behind during the Process 1 when after the first round only 50ms of execution is remaining and it has to wait a long time for that 50ms of execution. In these cases, the proposed algorithm is considered advantageous.

B. Example 2:

TABLE II.

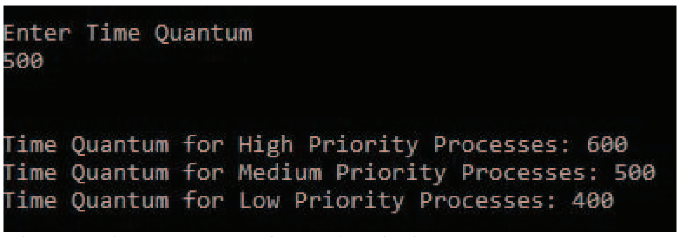
Process	Burst Time	Arrival Time	Priority
1	550	0	3
2	1250	0	1
3	1950	0	3
4	50	0	3
5	500	0	2
6	1200	0	1
7	100	0	3

Fig 2: Burst time, arrival time and priorities of processes

By the Round Robin algorithm: (Time Quantum=500)
Average Waiting Time = 2650.0 units
Average Turnaround Time = 3450.0
units Number of Context Switches = 14

By Improved Round Robin algorithm: (Time Quantum = 500)

Number of Context Switches = 12

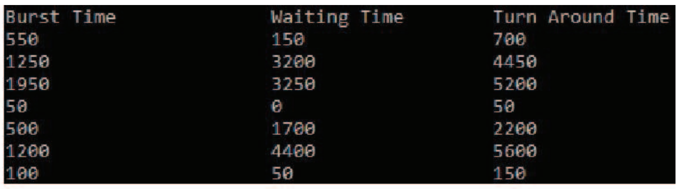


```
Enter Time Quantum
500


Time Quantum for High Priority Processes: 600
Time Quantum for Medium Priority Processes: 500
Time Quantum for Low Priority Processes: 400
```

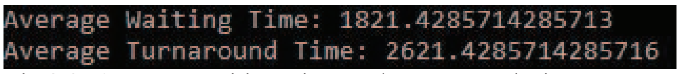
Fig 2.1: Time Quantum for each Priority

Results:



```
Burst Time          Waiting Time        Turn Around Time
550                 150                 700
1250                3200                4450
1950                3250                5200
50                  0                   50
500                 1700                2200
1200                4400                5600
100                 50                  150
```

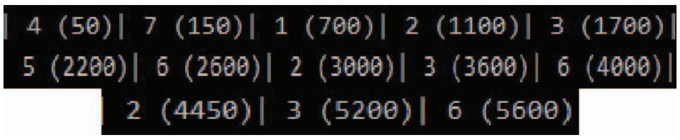
Fig 2.2: Individual Waiting time and Turnaround Time



```
Average Waiting Time: 1821.4285714285713
Average Turnaround Time: 2621.4285714285716
```

Fig 2.3: Average Waiting Time and Turnaround Time

Gantt Chart:

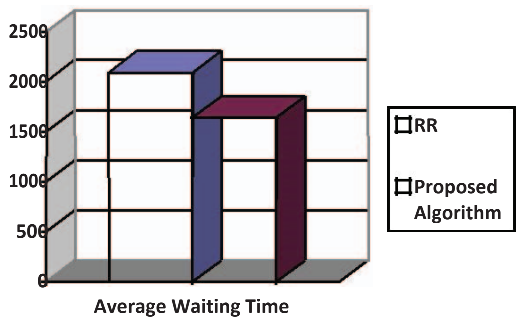


```
| 4 (50)| 7 (150)| 1 (700)| 2 (1100)| 3 (1700)|
 5 (2200)| 6 (2600)| 2 (3000)| 3 (3600)| 6 (4000)|
    | 2 (4450)| 3 (5200)| 6 (5600)
```

V. Observation

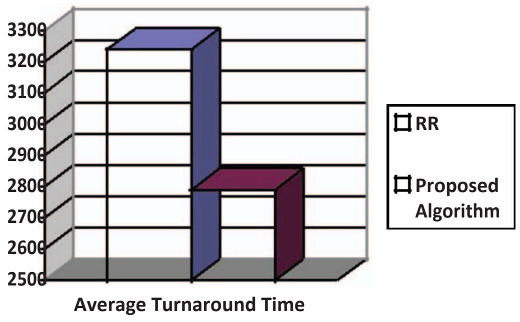
The graphs are used to compare the results of round robin algorithm with the proposed algorithm. Both the algorithms work on the same principle of providing a fair time slice to all the process. By introducing priority in round robin algorithm, the time quantum can be decided separately for each process and the CPU can be shared in a fairer way than in the existing algorithm. Context switches and average waiting, turnaround time have substantially decreased in the proposed algorithm.

Graphs for Example 1:



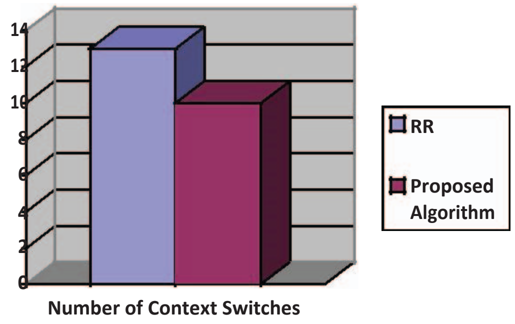
Graph 1.1: Comparison of average waiting time

Average waiting time is criteria of how long a process is in the ready queue [2]. 'Improved Round Robin' algorithm decreases these parameters to improve performance.



Graph 1.2: Comparison of average turnaround time

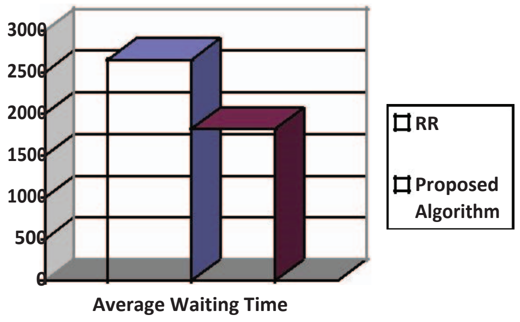
Average turnaround time determines when the process completes its execution [2]. 'Improved Round Robin' algorithm decreases the turnaround time of a process.



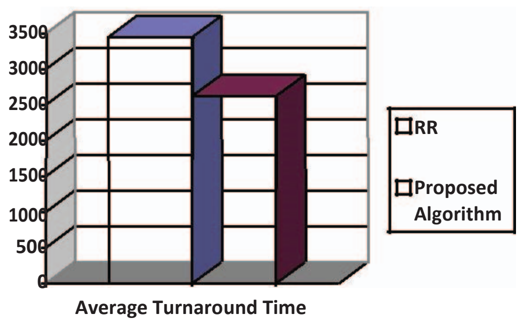
Graph 1.3: Comparison of number of context switches.

Number of context switches are reduced to decrease the overhead of saving and retrieving the PCBs.

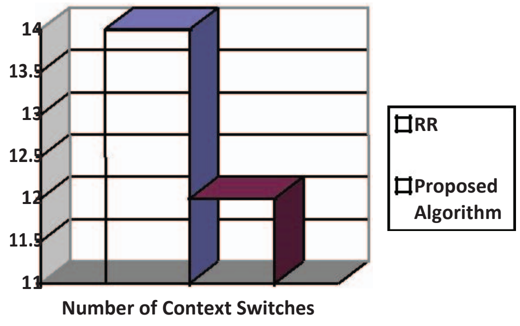
Graphs for Example 2:



Graph 2.1: Comparison of average waiting time



Graph 2.2: Comparison of average turnaround time



Graph 2.3: Comparison of number of context switches.

VI. Future Work

The algorithm can be operated within an algorithm of selecting the best scheduling algorithm dynamically i.e. based on the type of usage, different algorithms can be utilized together to give a better and efficient process scheduling. Other scheduling algorithms can also be used to modify the existing algorithm for effectiveness such as dynamically changing the time quantum, etc. The time quantum of the algorithm can be decided by the algorithm by considering the factors like average burst time, arrival time, etc. The algorithm can be implemented according to priority based algorithm by sorting the processes according to the priority.

References

[1] Alban Karapici, Enri Feka, Igli Tafa, Allkoçi "The Simulation of Round Robin and Priority Scheduling Algorithm", Information Technology - New Generations (ITNG), 2015 12th International Conference. 2015

[2] Silberschatz A., Galvin P., Gagne G, "Operating Systems Concepts", VIIIth Edition Wiley.

[3] Ahmed Alsheikhy, Reda Ammar and Raafat Elfouly, "An Improved Dynamic Round Robin Scheduling Algorithm Based on a Variant Quantum Time", Computer Engineering Conference (ICENCO), 2015 11th International, 2016.

[4] Malhar Thombare, Rajiv Sukhwani, Priyam Shah, Sheetal Chaudhari, Pooja Raundale, "Efficient Implementation of Multilevel Feedback Queue Scheduling", Wireless Communications, Signal Processing and Networking (WiSPNET), 2016.