HYBRID COMBINATION OF SJF AND ROUND ROBIN

JAHEER KHAN 211IT026 , SHUAIB JAWID 211IT087, BABAR SANKET 211IT015, ROUNAK JAIN 211IT055
INFORMATION TECHNOLOGY DEPARTMENT
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL

I.Abstract

One of the important roles played by CPU is process management. The CPU scheduling algorithms are essential for the performance of the system. The CPU utilization is maximized by the CPU scheduling algorithms. The algorithms used for scheduling CPU help in reducing the context switching, turnaround time, response time ,average waiting time. First Come First Serve, Round Robin, Shortest Job First and Priority Scheduling are some useful algorithms for scheduling CPU. The systems based on time sharing uses Round Robin CPU scheduling algorithm. This paper presents an algorithm which is hybrid of both Round Robin Algorithm and Shortest Job First(SJF) Algorithm in which the burst -time of shortest job is used as the time quantum. The modified algorithm is proven more useful than the conventional Round Robin. The hybrid of Round Robin and SJF algorithms have minimized the turnaround time and average waiting time to enhance CPU performance

II.Introduction:

Many processes are executed simultaneously by CPU in the multiprogramming systems. In order to maximize the CPU utilization, the CPU keeps on executing some process all the time so that it is not idle. This improves the efficiency of the CPU. The algorithms for scheduling CPU focuses on minimizing the turnaround -time, response -time and waiting -time and improving the performance of CPU. This paper throws light on modifying the Round Robin algorithm used for CPU Scheduling.

When a process is fetched from disk to memory then it is in a new state. In the ready queue, new state processes wait for the processor to be allocated. The interval for which the process waits for the processor to be allocated in the ready queue is called waiting-time. Interval between process submission to process completion is called turnaround time. The time required to complete a process or a task is called burst time.

The process gets the processor on the basis of CPU Scheduling Algorithms. The processor is allocated to the process on the basis of time on which they arrive in the First Come First Serve algorithm. The process which arrives first is given processor first. The understanding and implementation of this is easy. But, the average waiting time is high. In the Shortest Job First (SJF) algorithm, CPU is available for the processes having less burst time. The average waiting time is less. In the priority scheduling algorithm, the processes are assigned some priority. The process with maximum priority is allocated CPU first. In Round Robin (RR) Algorithm, every process is executed for a constant time interval called quantum. The process executes for a constant time interval and then the execution of another process takes place for a constant time interval and so on. In this paper, the Round Robin Algorithm for

scheduling CPU is modified and a hybrid of Round Robin and SJF algorithms is proposed so that the average waiting time is reduced.

III.Methodology

The hybrid scheduling algorithm combines the Round Robin (RR) and Shortest Job First (SJF) scheduling techniques to optimize process scheduling in operating systems. This algorithm aims to strike a balance between fairness and efficiency by considering the burst time of processes.

Steps involved in the hybrid scheduling algorithm:

1. Average Burst Time Calculation:

The first step is to calculate the average burst time for all processes. This is done by summing up the burst time of each process and dividing it by the total number of processes. The formula is as follows:

2. Determining Time Quantum:

Next, the algorithm determines the time quantum for the Round Robin scheduling. If the average burst time is greater than 2, the time quantum is set to the average burst time divided by the square of the number of processes with burst time greater than or equal to the average burst time. Otherwise, the time quantum is set to 2. The pseudo code for this step is as follows:

3. Main Scheduling Loop:

The algorithm enters a main scheduling loop that continues until all processes are executed. Within this loop, the algorithm selects the first process from the ready queue.

4. Dynamic RR Case:

If the burst time remaining for the selected process is greater than or equal to the average burst time, it is considered for Round Robin scheduling. If the burst time remaining is less than the time quantum, the process is considered complete. The completion time, waiting time, and turnaround time for the process are calculated and updated. If the burst time remaining is greater than the time quantum, the process is pre-empted, and the burst time remaining is reduced by the time quantum. The current time is updated, and the process is pushed back into the ready queue.

5. SJF Case:

If the burst time remaining for the process is less than the average burst time, it is considered for Shortest Job First scheduling. The algorithm searches for the process with the shortest burst time remaining among the processes in the ready queue. The shortest job is selected, and its completion time, waiting time, and turnaround time are calculated and updated. The process is marked as complete and removed from the ready queue. The current time is then updated.

The Hybrid scheduling algorithm includes the main scheduling loop, dynamic RR case, and SJF case. It involves updating process parameters, managing the ready queue, and checking for new process arrivals.

By employing this hybrid scheduling algorithm, operating systems can achieve a balance between fairness and efficiency in process scheduling. It considers the burst time of processes to make scheduling decisions, resulting in improved system performance.

IV.Working and Implementation

1. User Input: The user is prompted to enter the number of processes and the arrival time and burst time for each process.

2. Sorting: The processes are sorted based on their arrival time using the STL `sort` function.

3. Average Burst Time Calculation: The total burst time of all processes is calculated, and the average burst time is determined by dividing the total burst time by the number of processes.

Pseudo code for average burst time calculation

```
total_burstTime <- 0

num_processes <- n


for i <- 0 to n-1 do

    total_burstTime <- total_burstTime + processes[i].burstTime

end for


avg_burstTime <- total_burstTime / num_processes
```

4. Determining Time Quantum: The algorithm calculates the time quantum for the Round Robin scheduling algorithm based on the average burst time. If the average burst time is greater than 2, the time quantum is set to the average burst time divided by the square of the number of processes with burst time greater than or equal to the average burst time. Otherwise, the time quantum is set to 2.

Pseudo code for determining Time Quantum

```
sum_burstTime_RR <- 0

num_RR_processes <- 0


for i <- 0 to n-1 do

        if processes[i].burstTime >= avg_burstTime then

            sum_burstTime_RR <- sum_burstTime_RR + processes[i].burstTime

            num_RR_processes <- num_RR_processes + 1

   end if

end for


time_quantum <- 0


if (sum_burstTime_RR / (num_RR_processes * num_RR_processes)) > 2 then

   time_quantum <- sum_burstTime_RR / (num_RR_processes * num_RR_processes)

else

   time_quantum <- 2
```

end if

5. Hybrid Scheduling: The `hybrid_scheduling` function is called with the sorted processes array, the number of processes, the time quantum, and the average burst time. This function implements the hybrid scheduling algorithm. The ready queue is initialized with the first process (index 0), which arrived first. The process is marked as in the queue.

Pseudo code

```
function hybrid_scheduling(processes[], n, quantum, avg_burstTime)
    readyQueue <- empty queue
    readyQueue.push(0)
    processes[0].inQueue <- true
    currentTime <- 0
    programsExecuted <- 0
    updateQueue
end function
```

6. Main Scheduling Loop: The main scheduling loop continues until all processes are executed. Within the loop, the algorithm selects the first process in the ready queue.

7. Dynamic RR Case: If the burst time remaining for the process is greater than or equal to the average burst time, it is considered for Round Robin scheduling. If the burst time remaining is less than the time quantum, the process is considered complete. The completion time, waiting time, and turnaround time for the process are calculated and updated. If the burst time remaining is greater than the time quantum, the process is pre-empted, and the burst time remaining is reduced by the time quantum. The current time is updated, and the process is pushed back into the ready queue.

Pseudo code for RR case

```
int i = readyQueue.front()
if (processes[i].burstTimeRemaining >= avg_burstTime)
{
   readyQueue.pop()
    if (processes[i].burstTimeRemaining < quantum)
   {
      processes[i].isComplete = true;
                              currentTime += processes[i].burstTimeRemaining;
      programsExecuted++;
      processes[i].inQueue = false
      checkForNewArrival
   }
   else
   {
      processes[i].burstTimeRemaining -= quantum;
      currentTime += quantum
      checkForNewArrival
      readyQueue.push(i);
```

```
    }
}
```

8. SJF Case: If the burst time remaining for the process is less than the average burst time, it is considered for SJF scheduling. The algorithm searches for the process with the shortest burst time remaining among the processes in the ready queue. The shortest job is selected, and its completion time, waiting time, and turnaround time are calculated and updated. The process is marked as complete, and it is removed from the ready queue. The current time is then updated.

```
int shortest_job = readyQueue.front();

float shortest_time = processes[i].burstTimeRemaining;

for (int i = 0; i < n; ++i)

{

    if (processes[i].burstTimeRemaining < avg_burstTime && processes[i].burstTimeRemaining > 0 && processes[i].burstTimeRemaining < shortest_time && processes[i].inQueue)

    {s

        shortest_job = i;

                    shortest_time = processes[i].burstTimeRemaining;

    }

    if (processes[i].burstTimeRemaining >= avg_burstTime && processes[i].burstTimeRemaining > 0 && processes[i].inQueue)

    {

        break;

    }

}

processes[shortest_job].isComplete = true;

currentTime += processes[shortest_job].burstTimeRemaining;

programsExecuted+

remove(shortest_job, readyQueue

processes[i].inQueue = false

checkForNewArrival
```

9. Checking for New Arrivals: After executing a process, the algorithm checks for any new arrivals. If a process has arrived (its arrival time is less than or equal to the current time) and it is not already in the queue, it is marked as in the queue and added to the ready queue.

```
for (int i = 0; i < n; i++) {

    Process p = processes[i];

    if (p.arrivalTime <= currentTime && !p.inQueue && !p.isComplete)

    {

        processes[i].inQueue = true;

        readyQueue.push(i);

    }

}
```

V. Results

```
Enter the number of processes: 5
Enter arrival time and burst time of each process 1: 0 40

Enter arrival time and burst time of each process 2: 10 30

Enter arrival time and burst time of each process 3: 20 20

Enter arrival time and burst time of each process 4: 30 25

Enter arrival time and burst time of each process 5: 40 15
```

Fig 1: First data set

```
Average burst time= 26
Time quantum= 17.5
```

Fig 2: avg burst time and time quantum for hybrid algorithm

Time quantum= sum of burst times of processes having burst time > avg burst time divided by square of no. of these processes.

```
        The Process Status

Process ID      Arrival Time    Burst Time      Completion Time Turn Around Time        Waiting Time    Response Time
        1               0               40              105             105             65              0
        2               10              30              47.5            37.5            7.5             7.5
        3               20              20              82.5            62.5            42.5            42.5
        4               30              25              130             100             75              75
        5               40              15              62.5            22.5            7.5             7.5

        Average Waiting Time: 39.5
        Average Turn Around Time: 65.5
        Average Response Time: 26.5
```

Fig 2.1 Hybrid Algorithm result for dataset 1

```
enter the choice: 1
 P              AT              BT              FT              WT              RT              TAT
 1              0               40              40              0               0               40
 2              10              30              70              30              30              60
 3              20              20              90              50              50              70
 4              30              25              115             60              60              85
 5              40              15              130             75              75              90

Average waiting time = 43
Average turn around time = 69
Average response time = 43
```

Fig 2.2: FCFS (First Come First Serve) result for dataset 1

```
enter the choice: 2
 P              AT              BT              FT              WT              RT              TAT
 1              0               40              40              0               0               40
 2              10              30              130             90              90              120
 3              20              20              75              35              35              55
 4              30              25              100             45              45              70
 5              40              15              55              0               0               15

Average waiting time = 34
Average turn around time = 60
Average response time = 34
```

Fig 2.3: SJF (Shortest Job First) result for dataset 1

```
enter the choice: 3
 P              AT             BT             FT             WT             RT             TAT
 1              0              40             40             0              0              40
 2              10             30             130            90             90             120
 3              20             20             75             35             35             55
 4              30             25             100            45             45             70
 5              40             15             55             0              0              15

Average waiting time = 34
Average turn around time = 60
Average response time = 34
```

Fig 2.4: SRTF (Shortest Remaining Job First) result for dataset 1

```
enter the choice: 4

Enter the Time Slice or Quantum: 17
 P              AT             BT             FT             WT             RT             TAT
 1              0              40             119            79             0              119
 2              10             30             98             58             7              88
 3              20             20             122            82             31             102
 4              30             25             130            75             38             100
 5              40             15             113            58             58             73

Average waiting time = 70.4
Average turn around time = 96.4
Average response time = 26.8
```

Fig 2.5: RR (Round Robin) result for dataset 1

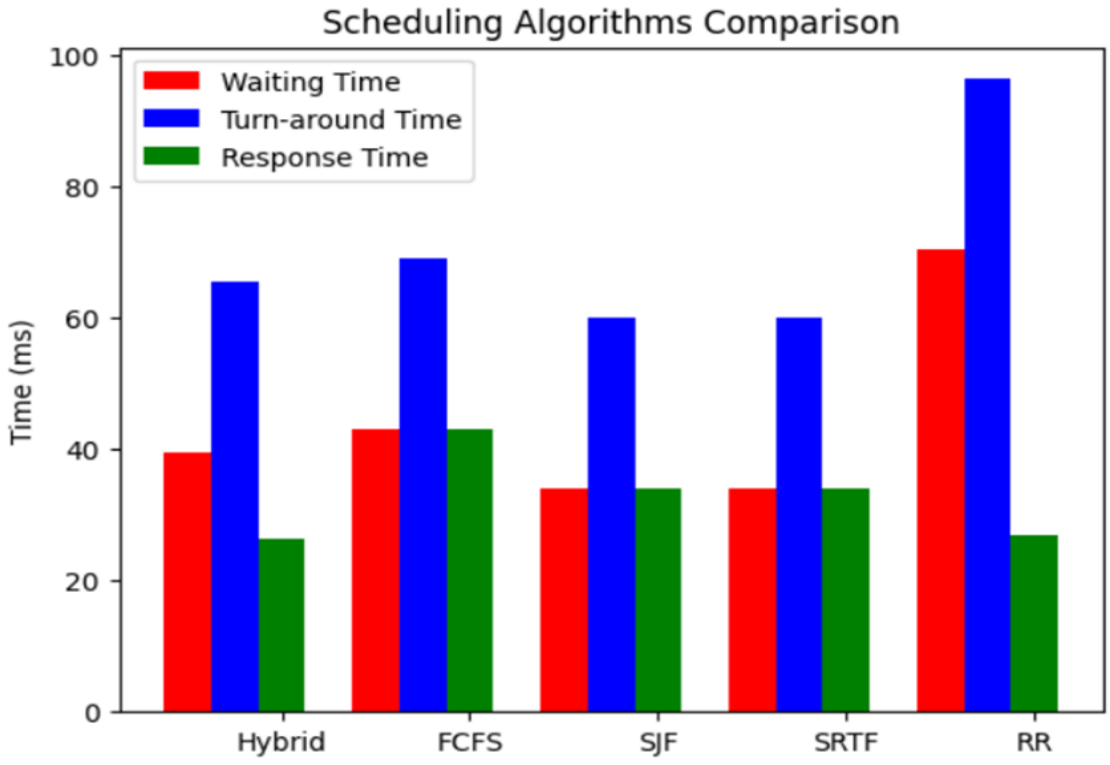

Fig3: First Experimentation Results using Dataset1

2nd Experimental Data Set:-

```
PS C:\Users\Rounak Jain\Desktop\it253 OS lab> cd "c:\Users\
Enter the number of processes: 5
Enter arrival time and burst time of each process 1: 0 12

Enter arrival time and burst time of each process 2: 5 14

Enter arrival time and burst time of each process 3: 10 1

Enter arrival time and burst time of each process 4: 12 3

Enter arrival time and burst time of each process 5: 15 11
```

Fig 4: Second data set

```
Average burst time= 8.2
Time quantum= 4.11111
```

Fig 4.1: avg burst time and time quantum for hybrid algorithm

```
   The Process Status

ess ID      Arrival Time    Burst Time      Completion Time Turn Around Time        Waiting Time    Response Time
   1               0               12              12              12              0               0
   2               5               14              34.1111         29.1111         15.1111         7
   3               10              1               17.1111         7.11111         6.11111         6.11111
   4               12              3               20.1111         8.11111         5.11111         5.11111
   5               15              11              41              26              15              5.11111

  Average Waiting Time: 8.26667
  Average Turn Around Time: 16.4667
  Average Response Time: 4.66667
```

Fig 5.1 Hybrid Algorithm result for dataset 2

```
enter the choice: 1
 P              AT              BT              FT              WT              RT              TAT
 1              0               12              12              0               0               12
 2              5               14              26              7               7               21
 3              10              1               27              16              16              17
 4              12              3               30              15              15              18
 5              15              11              41              15              15              26

 Average waiting time = 10.6
 Average turn around time = 18.8
 Average response time = 10.6
```

Fig 5.2: FCFS (First Come First Serve) result for dataset 2

```
enter the choice: 2
 P              AT             BT             FT             WT             RT             TAT
 1              0              12             12             0              0              12
 2              5              14             41             22             22             36
 3              10             1              13             2              2              3
 4              12             3              16             1              1              4
 5              15             11             27             1              1              12

Average waiting time = 5.2
Average turn around time = 13.4
Average response time = 5.2
```

Fig 5.3: SJF (Shortest Job First) result for dataset 2

```
 enter the choice: 3
  P              AT             BT             FT             WT             RT             TAT
  1              0              12             13             1              0              13
  2              5              14             41             22             22             36
  3              10             1              11             0              0              1
  4              12             3              16             1              1              4
  5              15             11             27             1              1              12

  Average waiting time = 5
  Average turn around time = 13.2
  Average response time = 4.8
```

Fig 5.4: SRTF (Shortest Remaining Job First) result for dataset 2

```
enter the choice: 4

Enter the Time Slice or Quantum: 4
 P              AT             BT             FT             WT             RT             TAT
 1              0              12             16             4              0              16
 2              5              14             38             19             3              33
 3              10             1              17             6              6              7
 4              12             3              20             5              5              8
 5              15             11             41             15             9              26

Average waiting time = 9.8
Average turn around time = 18
Average response time = 4.6
```

Fig 5.5: RR (Round Robin) result for dataset 2

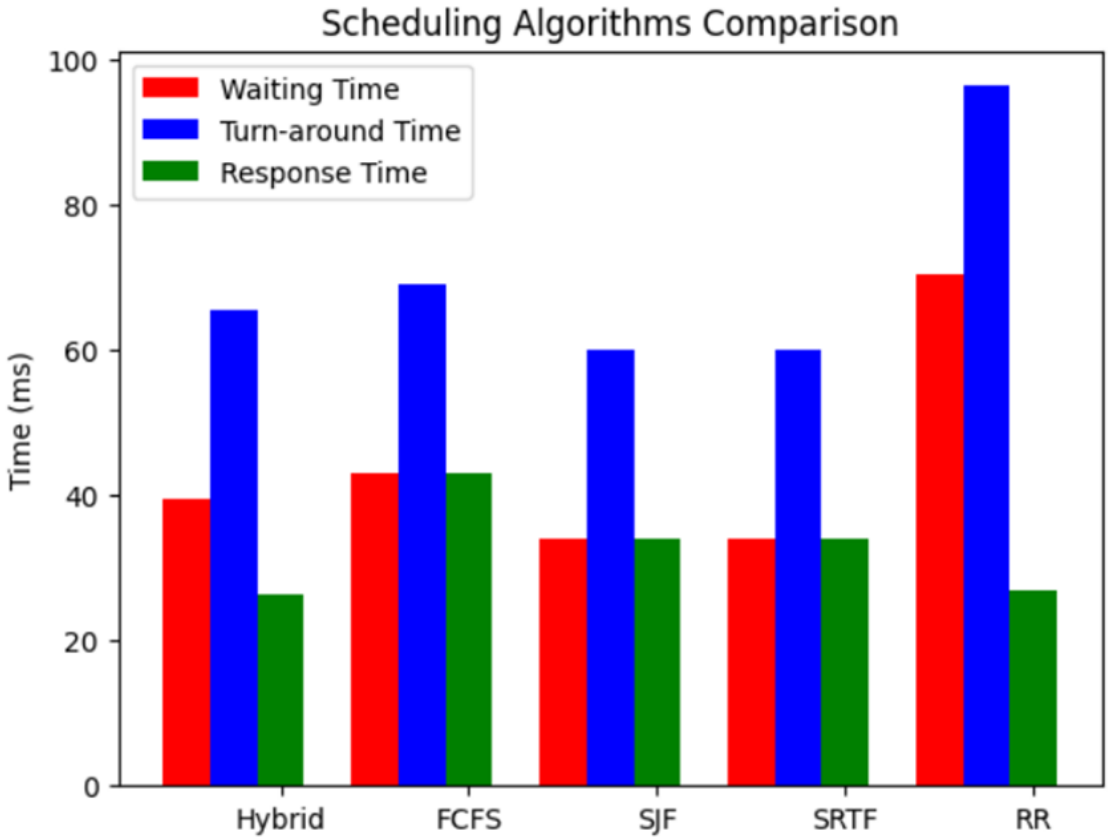

Fig 6: Second Experimentation Results using Dataset2

VI.Analysis

- From the *waiting time* perspective, SJF and SRTF algorithms have lower waiting times compared to the Hybrid algorithm. However, the Hybrid algorithm still performs better than the FCFS and RR algorithms.

- Regarding *turn-around time*, both SJF and SRTF algorithms have lower values than the Hybrid algorithm. Again, the Hybrid algorithm outperforms FCFS and RR algorithms in terms of turn-around time.

- Looking at the *response time*, we can see that the Hybrid algorithm has the lowest response time among all the algorithms. It performs significantly better than the FCFS and SJF algorithms and slightly better than the SRTF and RR algorithms.

Based on this analysis, we can conclude that the Hybrid algorithm demonstrates better performance overall compared to the other algorithms, especially in terms of response time. However, it is worth noting that different algorithms may excel in

different scenarios or under specific conditions, and a comprehensive evaluation should take into consideration other factors such as fairness, efficiency, and scalability before making a definitive conclusion.

VII.Conclusion and future trends

This study aimed to assess the performance of the proposed CPU scheduling algorithm under different process arrival scenarios. The algorithm categorizes processes based

on their burst times, applying SJF scheduling for shorter processes and RR scheduling for longer processes. Two distinct arrival orders were considered: small and medium processes arriving first, followed by large processes, and vice versa.

The findings of this evaluation shed light on the algorithm's behavior under varying process arrival conditions:

1. Small and Medium Processes First: When small and medium processes arrive first, the algorithm showcases enhanced performance in terms of turnaround time and response time for these processes. By prioritizing SJF for shorter processes, the algorithm efficiently minimizes the waiting time and maximizes CPU utilization for these tasks. Consequently, small and medium processes experience improved overall execution times compared to other standard scheduling algorithms.

2. Large Processes First: In scenarios where large processes arrive before smaller ones, the algorithm behaves similarly to SJF. As large processes are categorized as above-average burst time processes, they are scheduled using RR. However, since there are no shorter processes to utilize the RR time slices, the large processes effectively receive the entire time slice, resembling the behavior of pure SJF scheduling. As a result, large processes receive a fair share of CPU time, minimizing their waiting times and promoting efficient execution.

3. Flexibility in Handling Different Arrival Orders: The proposed algorithm exhibits adaptability to different process arrival orders. By combining SJF and RR, it ensures that both shorter and longer processes are treated appropriately, regardless of their arrival sequence. This adaptability contributes to improved scheduling efficiency and fairness, addressing the diverse burst time requirements of processes.

4 .Practical Applicability: The proposed algorithm shows promise for practical implementation in real-world systems. Its simplicity in combining two well-known scheduling techniques and its adaptability make it suitable for various operating environments, ranging from single-user systems to multi-user systems.

In conclusion, the proposed CPU scheduling algorithm, incorporating both SJF and RR techniques, demonstrates superior performance when small and medium processes arrive first. It optimizes resource utilization, reduces waiting times, and enhances turnaround and response times for these processes. Additionally, the algorithm showcases behavior similar to SJF when large processes arrive first, prioritizing their execution and minimizing waiting times. Overall, the algorithm exhibits flexibility in handling different process arrival orders, making it a promising choice for various real-world scenarios. Further

investigations could focus on evaluating its performance under different workload compositions and exploring strategies to handle process priorities or dynamic burst time changes.

VIII.Reference

1. Li, C., Wang, X., & Xie, Y. (2017). A Hybrid CPU Scheduling Algorithm Based on SJF and Round Robin. 2017 International Conference on Smart Grid and Electrical Automation (ICSGEA). https://doi.org/10.1109/ICSGEA.2017.00068

2. Yatish, D. (2014). A Comparative Analysis of CPU Scheduling Algorithms. International Journal of Advanced Research in Computer Science and Software Engineering, 4(3), 98-103.

3. Tanenbaum, A. S., & Bos, H. (2014). Modern Operating Systems (4th ed.). Pearson.

4. Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.

5. Hamdan, A. R., & Subramaniam, S. (2013). A Hybrid CPU Scheduling Algorithm Based on Round Robin and Shortest Job First. International Journal of Computer Applications, 75(12), 32-35. https://doi.org/10.5120/13197-4916

6. Dash, A. R., & Samantra, S. K. (2016). An optimized round Robin CPU scheduling algorithm with dynamic time quantum. International Journal of Computer Science, Engineering and Information.

7. G. Siva, Nageswara Rao, et al. (2014). "An Enhanced Dynamic Round Robin CPU Scheduling Algorithm." International Journal of Applied Engineering Research, Vol. 9, No. 15, pp. 3085-3098.

8. G. Siva, Nageswara Rao, et al. (2014). "Comparison of Round Robin CPU Scheduling Algorithm with Various Dynamic Time Quantum." International Journal of Applied Engineering Research, Vol. 9, No. 18, pp. 4905-4916.