VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

IMPROVED CPU SCHEDULING ALGORITHM FOR PROCESS MANAGEMENT BASED ON ROUND ROBIN ALGORITHM

OPERATING SYSTEMS (CSE 2005)

(J-COMPONENT)

Rushikesh Deotale (17BCE0294)

Kaushik(17BCE0398)

Kumar Harsh(17BCE0993)

Improved CPU Scheduling Algorithm for Process Management

ABSTRACT

The Round Robin (RR) CPU scheduling algorithm is a fair scheduling algorithm that gives equal time quantum to all processes. The choice of the time quantum is critical as it affects the algorithm's performance. We have suggested new algorithms and analyse how these algorithms work. We propose two new algorithms and compare them with existing round robin algorithm.
As the time quantum is static, it causes less context switching incase of high time quantum and high context switching incase of less time quantum. Increasing context switch leads to high avg. waiting time, high avg. turnaround time which is a overhead and degrades the system performance. So, the performance of the system solely depends upon the choice of optimal time quantum which is dynamic in nature.

We use two methods to calculate time quantum and analyse these methods for different cases and conclude by looking at the results. We look how these algorithms work for different process times and asses how both of them work. We propose these two improved round robin algorithms and show how much more efficient these are. We will show that our proposed algorithm performs better than the RR algorithm, by reducing context switching, average waiting and average turnaround time.

INTRODUCTION

Multiprogramming is one of the most important aspects of operating systems. It requires several processes to be kept simultaneously in memory, the aim of which is maximum CPU utilization. If these several processes in the memory are ready to run at the same time, the operating system must choose which one among them to run first. Making this decision is CPU scheduling. CPU scheduling is the basis of multiprogramming systems. It refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. It is made by the part of the operating system called the scheduler, using a CPU scheduling algorithm.

SCHEDULING ALGORITHMS

In the First-Come-First-Serve (FCFS) algorithm, the CPU is assigned immediately to that process which arrives first at the ready queue. In Shortest Job First (SJF) algorithm, process having shortest CPU burst time will execute first. If two processes having same burst time and arrival time, then FCFS procedure is followed. Priority scheduling algorithm, provides the priority to each process and selects the highest priority process from the ready queue. A small unit of time quantum is given to each process present in the ready queue in case of Round Robin (RR) algorithm which maintains the fairness factor. In RR scheduling, processes get fair share of CPU because of static time quantum assign to each process and the context switch is inversely proposnal to choice of static time quantum which degrades the overall performance of the system (high average waiting time & average turnaround time). This factor motivates us to design an improved algorithm which is able to increase the system performance by reducing the number of context switches, average waiting time & average turnaround time using the concept of dynamic time quantum.

PROPOSED ALGORITHMS

We are proposing two new CPU scheduling algorithm which have been derived from the existing algorithms. Both these algorithms change the time quantum for after each cycle.

The first one increases the time quantum by two after every cycle. The algorithm is as follows:

1. We read the time quantum and number of processes with their respective arrival time
2. Next we run one cycle of processes using round robin technique using the time quantum.
3. After one cycle we increase the time quantum by two times to the previous one.
4. We repeat step 2 and 3 until all processes are completed

The second one calculates the time quantum on the basis of remaining CPU burst time for each process after each cycle. The algorithm is as follows:

1. We read the time quantum and number of processes with their respective arrival time.
2. Next we run the one cycle using round robin with the time quantum.
3. After completion of one cycle we compute the time quantum by calculating the median value using the remaining CPU burst time.
4. We also find the highest remaining CPU burst time. The new time quantum is the mean of highest and median value of the remaining CPU burst times of all processes.
5. We repeat steps from 2 to 4 until all processes are completed.

We use these algorithms and compare them with existing round robin algorithm.

CODES FOR OUR IMPLEMENTATION

Normal Round Robin(NRR)

```
#include<stdio.h>

int main()
{
    int i, limit, total = 0, x, counter = 0, time_quantum;

    int wait_time = 0, turnaround_time = 0, arrival_time[20], burst_time[20],
temp[20];

    float average_wait_time, average_turnaround_time;
    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    x = limit;
    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Details of Process[%d]\n", i + 1);
        printf("Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time:\t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }

    printf("\nEnter Time Quantum:\t");
    scanf("%d", &time_quantum);
    printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");

    for(total = 0, i = 0; x != 0;)
    {
        if(temp[i] <= time_quantum && temp[i] > 0)
        {
            total = total + temp[i];
            temp[i] = 0;
            counter = 1;
        }
        else if(temp[i] > 0)
```

```
        {
            temp[i] = temp[i] - time_quantum;
            total = total + time_quantum;
        }
        if(temp[i] == 0 && counter == 1)
        {
            x--;
            printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i],
total - arrival_time[i], total - arrival_time[i] - burst_time[i]);
            wait_time = wait_time + total - arrival_time[i] - burst_time[i];
            turnaround_time = turnaround_time + total - arrival_time[i];
            counter = 0;
        }
        if(i == limit - 1)
        {
            i = 0;
        }
        else if(arrival_time[i + 1] <= total)
        {
            i++;
        }
        else
        {
            i = 0;
        }
    }
    average_wait_time = wait_time * 1.0 / limit;
    average_turnaround_time = turnaround_time * 1.0 / limit;
    printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
    printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
    return 0;
}
```

-Round Robin with increasing time quantum(RR-1)

```
#include<stdio.h>

int main()
{
    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[20], burst_time[20],
temp[20];

    float average_wait_time, average_turnaround_time;

    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    x = limit;

    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Details of Process[%d]\n", i + 1);
        printf("Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time:\t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }
    printf("\nEnter Time Quantum:\t");
    scanf("%d", &time_quantum);
    printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");

    for(total = 0, i = 0; x != 0;)
    {
        if(temp[i] <= time_quantum && temp[i] > 0)
        {
            total = total + temp[i];
            temp[i] = 0;
            counter = 1;
        }
        else if(temp[i] > 0)
        {
            temp[i] = temp[i] - time_quantum;
            total = total + time_quantum;
        }
        if(temp[i] == 0 && counter == 1)
```

```
        {
            x--;
            printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i], total -
arrival_time[i], total - arrival_time[i] - burst_time[i]);
            wait_time = wait_time + total - arrival_time[i] - burst_time[i];
            turnaround_time = turnaround_time + total - arrival_time[i];
            counter = 0;
        }
        if(i == limit - 1)
        {
            i = 0;
            time_quantum=time_quantum*2;
        }
        else if(arrival_time[i + 1] <= total)
        {
            i++;
        }
        else
        {
            i = 0;
        }
    }
    average_wait_time = wait_time * 1.0 / limit;
    average_turnaround_time = turnaround_time * 1.0 / limit;
    printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
    printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
    return 0;
}
```

-Round robin with calculated time quantum(RR-2)

```
#include<stdio.h>

int main()
{
    int i, limit, total = 0, x, counter = 0,
time_quantum,j,highest=0,mid,median,a;

    int wait_time = 0, turnaround_time = 0, arrival_time[20], burst_time[20],
temp[20],med[20];

    float average_wait_time, average_turnaround_time;
    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    x = limit;

    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Details of Process[%d]\n", i + 1);
        printf("Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time:\t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }
    printf("\nEnter Time Quantum:\t");
    scanf("%d", &time_quantum);
    printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");

    for(total = 0, i = 0; x != 0;)
    {
        if(temp[i] <= time_quantum && temp[i] > 0)
        {
            total = total + temp[i];
            temp[i] = 0;
            counter = 1;
        }
        else if(temp[i] > 0)
        {
```

```
            temp[i] = temp[i] - time_quantum;
            total = total + time_quantum;
        }
        if(temp[i] == 0 && counter == 1)
        {
            x--;
            printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i],
total - arrival_time[i], total - arrival_time[i] - burst_time[i]);
            wait_time = wait_time + total - arrival_time[i] - burst_time[i];
            turnaround_time = turnaround_time + total - arrival_time[i];
            counter = 0;
        }
        if(i == limit - 1)
        {
            i = 0;
            for(j=0;j<limit;j++)
            {
               if(highest<temp[j])
               {
                 highest=temp[j];
               }
            }
            for(j=0;j<limit;j++)
              med[j]=temp[j];
           for (k = 0; i < limit; ++i)
           {

            for (j = k + 1; j < limit; ++j)
            {

              if (med[k] > med[j])
              {

                a =  med[k];
                med[k] = med[j];
                med[j] = a;
              }
            }

           }
```

```
            if(limit%2==0)
            {

              mid=limit/2;
              median=med[mid];
              mid++;
              median=(median+med[mid])/2;
            }
            else
            {
              mid=limit/2;
              median=med[mid];
            }
            time_quantum=(highest + median)/2;
        }
        else if(arrival_time[i + 1] <= total)
        {
            i++;
        }
        else
        {
            i = 0;
        }
    }
    average_wait_time = wait_time * 1.0 / limit;
    average_turnaround_time = turnaround_time * 1.0 / limit;
    printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
    printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
    return 0;
}
```

ANALYSIS OF OUR ALGORITHMS WITH DATA SETS

DATA-1

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	8
P2	0	9
P3	0	10
P4	0	6

TIME QUANTUM=3

NRR OUTPUT

```
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     8

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     9

Enter Details of Process[3]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[4]
Arrival Time:   0
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[4]              6               24                      18
Process[1]              8               26                      18
Process[2]              9               29                      20
Process[3]              10              33                      23

Average Waiting Time:   19.750000
Avg Turnaround Time:    28.000000
```

RR-1 OUTPUT

```
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     8

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     9

Enter Details of Process[3]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[4]
Arrival Time:   0
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[2]              9               15                      6
Process[3]              10              25                      15
Process[4]              6               31                      25
Process[1]              8               33                      25

Average Waiting Time:   17.750000
Avg Turnaround Time:    26.000000
```

RR-2 OUTPUT

```
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     8

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     9

Enter Details of Process[3]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[4]
Arrival Time:   0
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[1]              8               17                      9
Process[2]              9               23                      14
Process[4]              6               32                      26
Process[3]              10              33                      23

Average Waiting Time:   18.000000
Avg Turnaround Time:    26.250000
```

DATA-2

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	10
P2	1	5
P3	2	7
P4	3	9

TIME QUANTUM =2

NRR OUTPUT

```
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[2]
Arrival Time:   1
Burst Time:     5

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     7

Enter Details of Process[4]
Arrival Time:   3
Burst Time:     9

Enter Time Quantum:     2

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[2]              5               18                      13
Process[3]              7               24                      17
Process[1]              10              30                      20
Process[4]              9               28                      19

Average Waiting Time:   17.250000
Avg Turnaround Time:    25.000000
```

RR-1 OUTPUT

```
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[2]
Arrival Time:   1
Burst Time:     5

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     7

Enter Details of Process[4]
Arrival Time:   3
Burst Time:     9

Enter Time Quantum:     2

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[2]              5               8                       3
Process[3]              7               14                      7
Process[4]              9               22                      13
Process[1]              10              31                      21

Average Waiting Time:   11.000000
Avg Turnaround Time:    18.750000
```

RR-2 OUTPUT

```
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[2]
Arrival Time:   1
Burst Time:     5

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     7

Enter Details of Process[4]
Arrival Time:   3
Burst Time:     9

Enter Time Quantum:     2

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[2]              5               17                      12
Process[3]              7               21                      14
Process[4]              9               27                      18
Process[1]              10              31                      21

Average Waiting Time:   16.250000
Avg Turnaround Time:    24.000000
```

DATA-3

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	12
P2	0	13
P3	2	10
P4	2	9
P5	3	11
P6	3	7
P7	4	6

TIME QUANTUM=3

NRR OUTPUT

```
Enter Total Number of Processes:        7

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     12

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     13

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     10

Enter Details of Process[4]
Arrival Time:   2
Burst Time:     9

Enter Details of Process[5]
Arrival Time:   3
Burst Time:     11

Enter Details of Process[6]
Arrival Time:   3
Burst Time:     7

Enter Details of Process[7]
Arrival Time:   4
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[7]              6               38                      32
Process[4]              9               52                      43
Process[6]              7               55                      48
Process[1]              12              61                      49
Process[3]              10              63                      53
Process[5]              11              64                      53
Process[2]              13              68                      55

Average Waiting Time:   47.571430
Avg Turnaround Time:    57.285713
```

RR-1 OUTPUT

```
Enter Total Number of Processes:        7

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     12

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     13

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     10

Enter Details of Process[4]
Arrival Time:   2
Burst Time:     9

Enter Details of Process[5]
Arrival Time:   3
Burst Time:     11

Enter Details of Process[6]
Arrival Time:   3
Burst Time:     7

Enter Details of Process[7]
Arrival Time:   4
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[3]              10              26                      16
Process[4]              9               35                      26
Process[5]              11              45                      34
Process[6]              7               52                      45
Process[7]              6               57                      51
Process[1]              12              67                      55
Process[2]              13              68                      55

Average Waiting Time:   40.285713
Avg Turnaround Time:    50.000000
```

RR-2 OUTPUT

```
Enter Total Number of Processes:        7

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     12

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     13

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     10

Enter Details of Process[4]
Arrival Time:   2
Burst Time:     9

Enter Details of Process[5]
Arrival Time:   3
Burst Time:     11

Enter Details of Process[6]
Arrival Time:   3
Burst Time:     7

Enter Details of Process[7]
Arrival Time:   4
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[3]              10              42                      32
Process[4]              9               48                      39
Process[5]              11              55                      44
Process[6]              7               59                      52
Process[7]              6               61                      55
Process[1]              12              66                      54
Process[2]              13              68                      55

Average Waiting Time:   47.285713
Avg Turnaround Time:    57.000000
```

DATA-4

PROCESS	ARRIVAL TIME	BURST TIME
P1	0	10
P2	0	11
P3	1	7
P4	2	14
P5	3	9
P6	4	8

TIME QUANTUM=5

NRR OUTPUT

```
Enter Total Number of Processes:        6

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     11

Enter Details of Process[3]
Arrival Time:   1
Burst Time:     7

Enter Details of Process[4]
Arrival Time:   2
Burst Time:     14

Enter Details of Process[5]
Arrival Time:   3
Burst Time:     9

Enter Details of Process[6]
Arrival Time:   4
Burst Time:     8

Enter Time Quantum:     5

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[1]              10              35                      25
Process[3]              7               41                      34
Process[5]              9               48                      39
Process[6]              8               50                      42
Process[2]              11              55                      44
Process[4]              14              57                      43

Average Waiting Time:   37.833332
Avg Turnaround Time:    47.666668
```

RR-1 OUTPUT

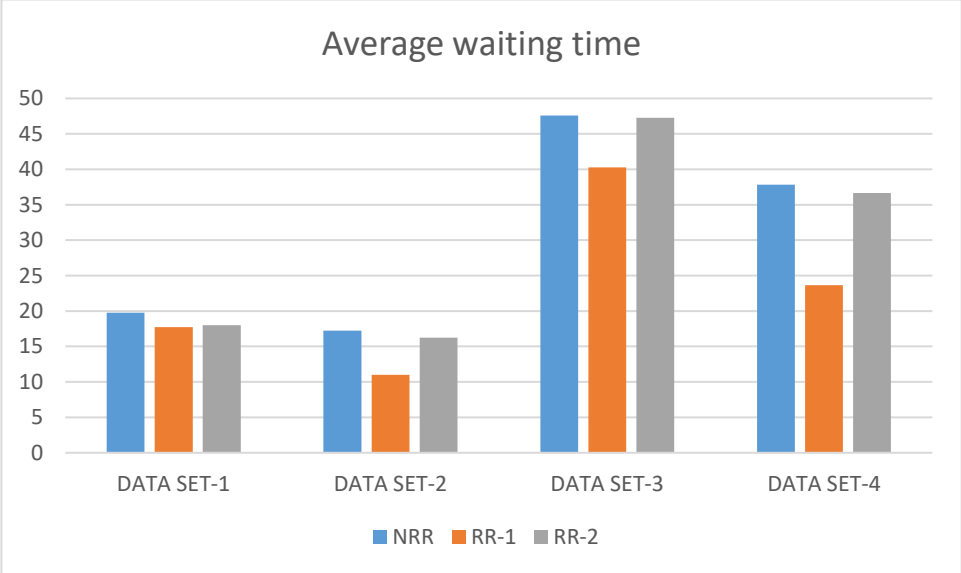
```
Enter Total Number of Processes:        6

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     11

Enter Details of Process[3]
Arrival Time:   1
Burst Time:     7

Enter Details of Process[4]
Arrival Time:   2
Burst Time:     14

Enter Details of Process[5]
Arrival Time:   3
Burst Time:     9

Enter Details of Process[6]
Arrival Time:   4
Burst Time:     8

Enter Time Quantum:     5

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[1]              10              10                      0
Process[2]              11              21                      10
Process[3]              7               27                      20
Process[4]              14              40                      26
Process[5]              9               48                      39
Process[6]              8               55                      47

Average Waiting Time:   23.666666
Avg Turnaround Time:    33.500000
```

RR-2 OUTPUT

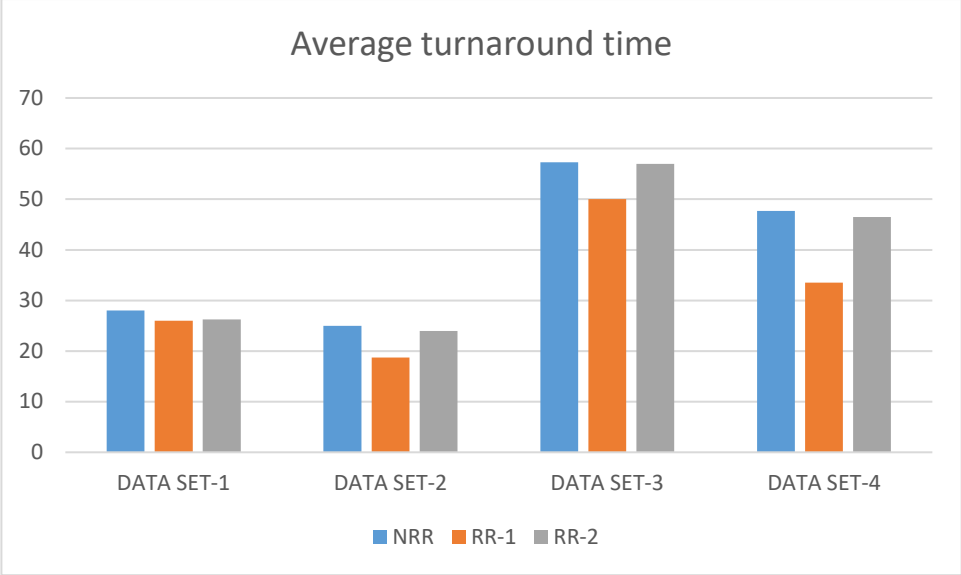
```
Enter Total Number of Processes:        6

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     10

Enter Details of Process[2]
Arrival Time:   0
Burst Time:     11

Enter Details of Process[3]
Arrival Time:   1
Burst Time:     7

Enter Details of Process[4]
Arrival Time:   2
Burst Time:     14

Enter Details of Process[5]
Arrival Time:   3
Burst Time:     9

Enter Details of Process[6]
Arrival Time:   4
Burst Time:     8

Enter Time Quantum:     5

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[1]              10              35                      25
Process[2]              11              41                      30
Process[3]              7               42                      35
Process[5]              9               51                      42
Process[6]              8               53                      45
Process[4]              14              57                      43

Average Waiting Time:   36.666668
Avg Turnaround Time:    46.500000
```

CONCLUSION

In all our results, our proposed algorithms gave better results. We used waiting time and turnaround time as our parameters to compare our algorithms on the four data sets we used. We also calculated average waiting time and average turnaround time to measure our results for overall processes. We conclude that the algorithms proposed worked better than the normal round robin algorithm.
We used two new methods which can improve the existing round robin. The new methods are equally more efficient than the existing round robin method. For some of our data the second round robin method performed which calculated the round robin performed better while in other cases the first method in which we increased the time quantum simply irrespective of remaining CPU burst time performed better.
We can conclude that one can use any one of our algorithms according to their needs and requirements.



AVERAGE WAITING TIME



Average Turnaround time

FUTURE WORK

These algorithms can be implemented in different way for increasing efficiency of CPU scheduling. Our algorithms give better results in helping CPU scheduling improvement. These algorithms can also be helpful in research in this field. Also these algorithms promise for more scope of improvement in CPU scheduling.

REFERENCES

1. Noon, A., Kalakech, A., & Kadry, S. (2011). A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average. *arXiv preprint arXiv:1111.5348*.

2. Rakesh MohantyManas Das, M. Lakshmi Prasanna, Sudhashree "Design and Performance Evaluation of A New Proposed Fittest Job First Dynamic Round Robin (FJFDRR) Scheduling Algorithm" International Journal of Computer Information Systems, Vol. 2, No. 2, 2011.

3. Operating System Concepts ,8th Ed.,Abraham Silberschatz, Peter B. Galvin, Grege Gagne . ISBN 978- 81-265-2051-0.

4. Singh, A., Goyal, P., & Batra, S. (2010). An optimized round robin scheduling algorithm for CPU scheduling. *International Journal on Computer Science and Engineering*, *2*(07), 2383-2385.

5. Mishra, M. K. (2012). An improved round robin CPU scheduling algorithm. Journal of Global Research in Computer Science, 3(6), 64-69.