Volume 3, No. 3, March 2012

Journal of Global Research in Computer Science

RESEARCH PAPER

Available Online at www.jgrcs.info

ISSN-2229-371X

ENHANCING THE CPU PERFORMANCE USING A MODIFIED MEAN-DEVIATION ROUND ROBIN SCHEDULING ALGORITHM FOR REAL TIME SYSTEMS.

H.s.Behera[*1], Sreelipa Curtis[2], bijayalaxmi panda[3]

*1 Department of Computer Science and Engineering, Veer Surendra Sai University of Technology,
Burla, Sambalpur, Odisha, India
hsbehera_india@gmail.com[1]

[2] Department of Computer Science and Engineering, Veer Surendra Sai University of Technology,
Burla, Sambalpur, Odisha, India
csrilipa@gmail.com[2]

[3] Department of Computer Science and Engineering, Veer Surendra Sai University of Technology,
Burla, Sambalpur, Odisha, India
bijayafouru@gmail.com[3]

Abstract: CPU scheduling is the basis of multiprogrammed operating systems. Round Robin CPU scheduling algorithm was considered as the optimized CPU scheduling algorithm when compared with the traditional FCFS, SJF and Priority scheduling algorithm. But with the elapse of time, the RR scheduling algorithm was further optimized by using extended or combination of CPU scheduling algorithms to enhance the efficiency of the CPU. By switching the CPU processes, the operating system can make the computer more productive and therefore choosing an optimized and efficient time quantum is a very important factor. This paper presents a new CPU scheduling algorithm coined Enhancing CPU scheduling using a modified mean deviation round robin (MMDRR) scheduling algorithm for real time system. MMDRR is experimentally proven better than traditional RR, SMDRR and SRBRR by reducing the context switches, average waiting time and average turnaround time significantly.

Keywords: Scheduling, round robin, burst time, waiting time, turnaround time.

INTRODUCTION

Basics:

A computer system consists of four components: the hardware, the operating system, the application programs and the users. The hardware – the CPU, the memory, and the I/O devices – provides the basic computing resources for the system. To enhance the performance of the CPU, day by day different scheduling algorithms are developed. An operating system is a program that manages the computer hardware.

Scheduling algorithm:

The FCFS (First Come First Serve) algorithm being the first refers that any process that arrives first in the ready queue is processed first. The second being the SJF (Shortest Job First) scheduling algorithm where the process having the shortest burst time is processed first. Following it the priority scheduling algorithm was developed where each process was assigned a priority and accordingly the process where executed. In case of priority scheduling, priority is assigned to each process and CPU is allocated to the process with highest priority. Equal priority processes are scheduled in FCFS order. But the development of the Round Robin scheduling algorithm was considered as the optimized algorithm where a fixed slice of time quantum was chosen and hence the processes were executed accordingly.

Motivation:

In RR scheduling a fixed time quantum is given to all process that are awaits in ready queue for execution. So the chance of frequent switches between processes increases by which efficiency of CPU decreases. On the other hand if the time slice considered is a large one then waiting time and turnaround time increases. In order to overcome these above situations, we have proposed an algorithm that uses a mean deviation dynamic time quantum concept.

Related Work:

Many research works has been done under this topic to enhance the performance of CPU. The static time quantum which is a limitation of RR was removed by taking dynamic time quantum by Matarneh [3]. SRBRR algorithm [1] uses a new approach that it is using dynamic time quantum in which time quantum repeatedly changes with each cycle of execution. SMDRR algorithm [2] is based on dynamic time quantum where we use the subcontrary mean or harmonic mean to find the time quantum for the processes to execute.

Our Contribution:

In this paper, the main objective is to reduce average waiting time and turnaround time as compared with the RR scheduling algorithm, SRBRR scheduling algorithm and the SMDRR scheduling algorithm. For this purpose, we have developed an algorithm that drastically reduces average waiting time and turnaround time.

Organization of Paper:

This paper represents a method for reducing context switching, average waiting time and average turnaround time using random sorting and dynamic quantum with burst task component and priority task component. Section 2 describes all preliminary work. Section 3 presents proposed approaches. Section 4 shows experimental analysis and comparison of result. In Section 5 conclusion is given.

BACKGROUND WORK

Terminology:

A program in execution is called a process. The processes, waiting to be assigned to a processor are put in a queue called ready queue. The performance of the CPU mainly depends upon many factors such as CPU utilization, Throughput, Turnaround Time (TAT), Waiting Time (WT), Context Switch (CS), Response Time etc. The utilization of the CPU is called CPU utilization where we keep the CPU as busy as possible. The number of processes completed per unit time is called Throughput. Waiting Time is the sum of the periods spent waiting in the ready queue. Time from the submission of a request until the first response is produced is called Response Time. Turnaround Time is the interval from the time of the submission of a process to the time of completion is the turnaround time. Context switch is the number of times the process switches to get execute. Scheduler selects a process from queues in some manner for its execution. In non-preemption, CPU is assigned to a process; it holds the CPU till its execution is completed. But in preemption, running process is forced to release the CPU by the newly arrived process. In time sharing system, the CPU executes multiple processes by switching among them very fast. The number of times CPU switches from one process to another is called as the number of context switches.

RR Scheduling Algorithm:

In RR, each ready task runs turn by in turn in a cyclic queue for limited time slices. It is widely used in traditional OS. RR is a hybrid model i.e. clock driven model (e.g. cyclic model) as well as event driven (e.g. Preemption). The performance of RR algorithm is highly dependent on time slice. For low time-slice context switching is more and for high time-slice response time is more. So the time quantum plays most determining factor for the performance of RR algorithm.

SRBRR Algorithm:

In Shortest Remaining Burst Round Robin algorithm, the time quantum is taken as the median of the increasingly sorted burst time of all the processes. The jobs are sorted in ascending order of their burst time. The time slice chosen is dynamic time quantum where the time quantum is repeatedly adjusted according to the remaining burst time of currently running processes. To get the optimal time quantum, median of the burst time is taken as the time quantum.

SMDRR Scheduling Algorithm:

In subcontrary mean dynamic round robin scheduling algorithm, the time quantum is taken as the subcontrary or harmonic mean of the increasingly sorted burst time of all the processes and this change dynamically in every cycle till the end of processes.

PROPOSED APPROACH:

In our proposed algorithm the time slice taken is the summation of mean and variance of the increasingly sorted burst time of all the processes.

Uniqueness of our approach:

In our algorithm, the jobs are sorted in ascending order of their burst time to give better turnaround time and waiting time. Performance of RR algorithm solely depends upon the size of time quantum. If it is very small, it causes too many context switches. If it is very large, the algorithm degenerates to FCFS. So our algorithm solves this problem by taking a dynamic time quantum where the time quantum is repeatedly adjusted according to the remaining burst time of currently running processes. To get the optimal time quantum, the summation of mean and standard deviation of the burst time is taken as the time quantum.

Our proposed approach:

In the proposed algorithm, when processes are already present in the ready queue, their arrival time is assigned to zero before they are allocated to the CPU. The burst time and the number of processes (n) are accepted as input with counter value 'i'.

Let TQ be the time quantum. The TQ calculated is the summation of mean and standard deviation. So, the TQ is calculated by the following formulae (4) as follows:

Mean = $\mu_{\mathbf{x}} = 1/n\ (x_1 + x_2 + \ldots. + x_n)$ (1)

Variance = $\boldsymbol{\sigma}^2 = (1/n)\ \boldsymbol{\Sigma}\ (x_i - \mu x)^2$ (2)

Deviation = $\boldsymbol{\sigma} = \{(1/n)\ \boldsymbol{\Sigma}\ (x_i - \mu x)^2\}^{1/2}$ (3)

TQ = Mean + Standard Deviation

$= \mu x + \sigma$ (4)

Where n = Total no of processes
x = Set of processes
and $(x_1, x_2, \ldots.., x_n) \in X$

Time quantum is assigned to each process. The process having the shortest job is allocated to the CPU. Time quantum is recalculated with remaining burst time after each execution of each cycle.

Pseudocode of the proposed algorithm:

```
1.      First all the processes present in ready queue are
        sorted in ascending order of their burst time.
        n ?  number of processes
        i ?  counter value
2.      While(RQ!=NULL)
        //TQ = Time Quantum = Mean + Deviation
        (remaining burst time of all the processes)
        TQ = ((1/n) (x_1 + x_2 + .... + x_n))
                            + {((1/n)  Σ  (x_i  -  ı'
                  x)^2)}^{1/2}
        // n = Total no. of processes
        // X = Set of processes ,where (x_1,x_2.....x_n) ?X
        //RQ = Ready Queue
        //TQ = Time Quantum
        //if TQ> maximum burst time, then max ( BT) ?
TQ
3.      Assign TQ to (1 to n) process
        for i=1 to n
        {
                Pi ?  TQ
                //P_i = Process i
        }
        end for
        // Assign TQ to all the available processes and the
        processes having the least burst time are allocated
        to the CPU first for execution.
4.      Calculate the remaining burst time of the processes.
5.      if ( new process is arrived and BT != 0 )
        //BT = burst time
        go to step 1
        else if ( new process is not arrived and BT != 0 )
        go to step 2
        else if ( new process is arrived and BT == 0)
        go to step 1
        else
        go to step 6
        end if
        end while
6.      Calculate ATT, AWT and CS.
        //ATT = Average Turnaround Time
        //AWT= Average Waiting Time
        //CS = number of Context Switches
7.      End
```

Flow chart of the proposed algorithm:

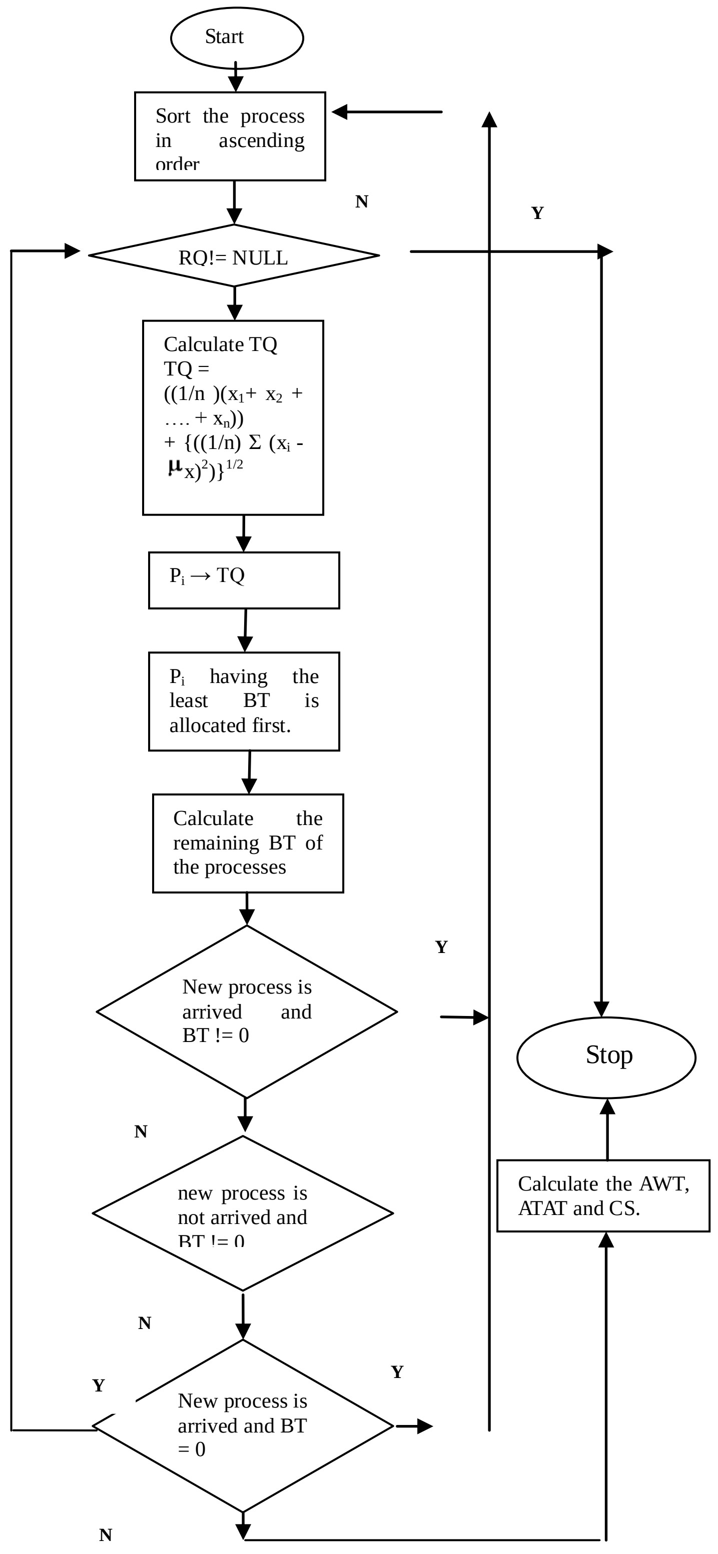
Start
Sort the process in ascending order
RQ!= NULL
N
Y
Calculate TQ
TQ = ((1/n)(x1+ x2 + …. + xn)) + {((1/n) Σ (xi - μx)²)}^1/2
Pi → TQ
Pi having the least BT is allocated first.
Calculate the remaining BT of the processes
New process is arrived and BT != 0
Y
N
new process is not arrived and BT != 0
N
New process is arrived and BT = 0
Y
Y
N
Stop
Calculate the AWT, ATAT and CS.

Illustration:

Given the burst time sequence of the processes as P1= 13, P2= 35, P3= 46, P4= 63, P5= 97. Initially the burst time of all the processes were sorted in ascending order which resulted in sequence P1, P2, P3, P4 and P5. Then the mean of the above burst time which was calculated to be 51 and standard deviation to be 31. The summation of mean and standard deviation was calculated to be 82 and thus was assigned as the time quantum for all the processes. In the next step remaining burst time of each process was calculated after allocating the time quantum. After first cycle the remaining burst time sequence for above processes changed to P5=15. When a process completes its burst time, it gets deleted from the ready queue automatically. So in this case, the processes P1, P2, P3 and P4 were deleted from the ready queue. The present remaining burst time were sorted in increasing order and then the summation of mean and standard deviation of the remaining burst time was assigned as the time quantum where we get 15 as the time quantum for the second cycle. The time quantum 15 was assigned to the process P5 for execution. This is how the processes are executed in the ready queue. The above process was continued till all the processes were deleted from the ready queue.

EXPERIMENTAL ANALYSIS

Assumptions:

The environment where all the experiments are performed is a single processor environment and all the processes are independent. There is equal priority given to all the processes. All the attributes like burst time, number of processes and the time slice of all the processes are known before submitting the processes to the processor. All processes are CPU bound. No processes are I/O bound. Also, a large number of processes is assumed in the ready queue for better efficiency. Since, the cases are assumed to be close to ideal, the Context Switching Time is equal to zero i.e. there is no Context Switch Overhead incurred in switching from one process to another. The TQ is taken in milliseconds (ms).

Experimental Frame Work:

For the performance evaluation of the proposed scheduling algorithm, our experiment consists of several input and output parameters. The input parameters consist of burst time, arrival time, time quantum and the number of processes. The output parameters consist of average waiting time, average turnaround time and number of context switches.

Data Set:

Two cases were considered for the experiment evaluation. Case-1 is for processes with zero arrival time. Case-2 is for processes with certain arrival time. In both case-1 and case-2, there are 3-subcases i.e. processes are taken in ascending, descending and random order. In each case, we have compared the experimental results of our proposed algorithm with the SRBRR scheduling algorithm, the SMDRR scheduling algorithm and the traditional round robin scheduling algorithm with fixed time quantum Q.

Performance Parameters:

The significance of our performance parameters for experimental analysis is as follows:

Average Waiting time (AWT): For the better performance of the scheduling algorithm, average waiting time of the processes should be less.

Average Turnaround time (ATAT): For the better performance of the scheduling algorithm, average turnaround time of the processes should be less.

Number of Context Switches (CS): For the better performance of the scheduling algorithm, the number of context switches should be less.

Experiments Performed:

To evaluate the efficiency of our proposed algorithm (MMDRR), the output parameters are compared with round robin (RR) scheduling algorithm, Shortest Remaining Burst Round Robin (SRBRR) scheduling algorithm and the subcontrary mean dynamic round robin (SMDRR) scheduling algorithm. This algorithm can work effectively with large number of processes. For simplicity we have taken five processes with ascending, descending and random order to illustrate our proposed algorithm. Here we have assumed a constant time quantum TQ equal to 25 ms in all the cases for RR scheduling algorithm.

CASE 1: With Zero Arrival Time

Increasing Order:

Five process P1, P2, P3, P4, P5 arriving at time 0 with burst time 13, 35, 46, 63, 97 respectively of each process shown in table 4.6.1 have been considered. Table 4.6.2 shows the comparing result of RR, SRBRR, SMDRR and our proposed algorithm (MMDRR).

Table 4.6.1 Data in increasing order

Processes	Arrival Time	Burst Time
P1	0	13
P2	0	35
P3	0	46
P4	0	63
P5	0	97

Table 4.6.2 Comparison among RR, SRBRR, SMDRR and MMDRR

Algorithms	QT	AWT	ATAT	CS
RR	25	97.4	148.2	11
SRBRR	46, 34, 17	71.6	122.4	7
SMDRR	36, 6, 15, 15,28	108.6	159.4	14
MMDRR	82,15	62.4	113.2	5

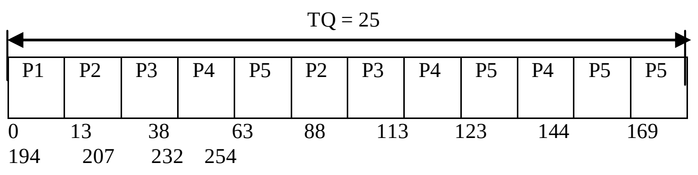

Figure. 4.6.2.1: Gantt chart for RR in Table 4.6.2

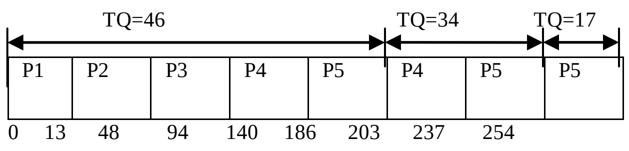

Figure. 4.6.2.2: Gantt chart for SRBRR in Table 4.6.2

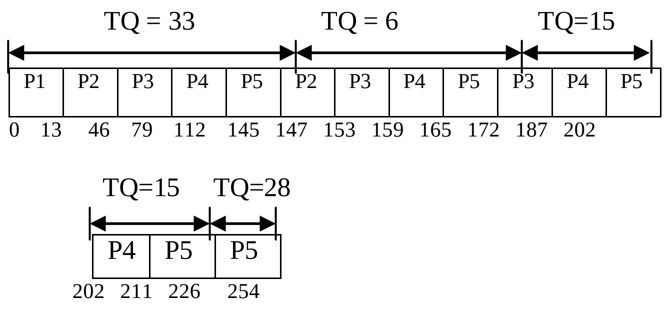

Figure. 4.6.2.3: Gantt chart for SMDRR in Table 4.6.2

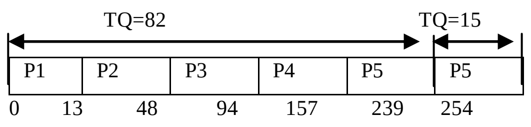

Figure. 4.6.2.4: Gantt chart for MMDRR in Table 4.6.2

Decreasing Order:

Five process P1, P2, P3, P4, P5 arriving at time 0 with burst time 86, 53, 32, 21, 9 respectively of each process shown in table 4.6.3 have been considered. Table 4.6.4 shows the comparing result of RR, SRBRR, SMDRR and our proposed algorithm (MMDRR).

Table 4.6.3 Data in decreasing order

Processes	Arrival Time	Burst Time
P1	0	86
P2	0	53
P3	0	32
P4	0	21
P5	0	9

Table 4.6.4 Comparison among RR, SRBRR, SMDRR and MMDRR

Algorithms	QT	AWT	ATAT	CS
RR	25	110.5	150.8	10
SRBRR	32, 38, 16	49.6	89.8	7
SMDRR	23, 19, 18, 26	60.8	101	10
MMDRR	70,16	43.2	83.4	5

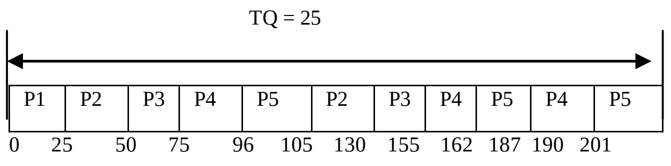

Figure. 4.6.4.1: Gantt chart for RR in Table 4.6.4

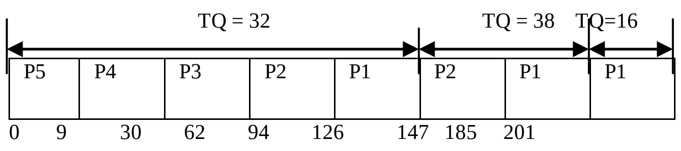

Figure. 4.6.4.2: Gantt chart for SRBRR in Table 4.6.4

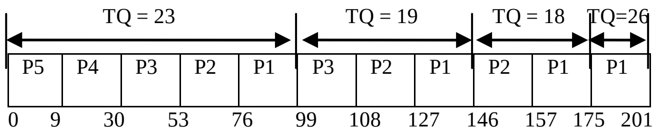

Figure. 4.6.4.3: Gantt chart for SMDRR in Table 4.6.4

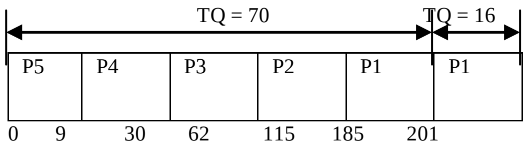

Figure. 4.6.4.4: Gantt chart for MMDRR in Table 4.6.4

Random Order:

Five process P1, P2, P3, P4, P5 arriving at time 0 with burst time 54, 99, 5, 27, 32 respectively of each process shown in table 4.6.5 have been considered. Table 4.6.6 shows the comparing result of RR, SRBRR, SMDRR and our proposed algorithm (MMDRR).

Table 4.6.5 Data in random order

Processes	Arrival Time	Burst Time
P1	0	54
P2	0	99
P3	0	5
P4	0	27
P5	0	32

Table 4.6.6 Comparison among RR, SRBRR, SMDRR and MMDRR

Algorithms	**QT**	**AWT**	**ATAT**	**CS**
RR	25	108.8	152.2	11
SRBRR	32, 45, 22	50.2	93.6	7
SMDRR	17, 29, 27, 35	99.2	142.6	11
MMDRR	79, 20	43.8	87.2	5

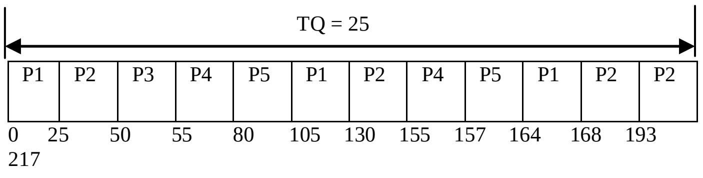

Figure. 4.6.6.1: Gantt chart for RR in Table 4.6.6

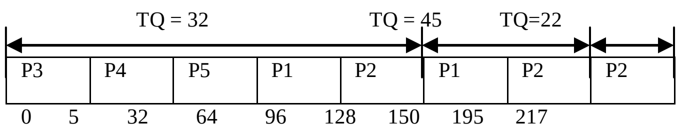

Figure. 4.6.6.2: Gantt chart for SRBRR in Table 4.6.6

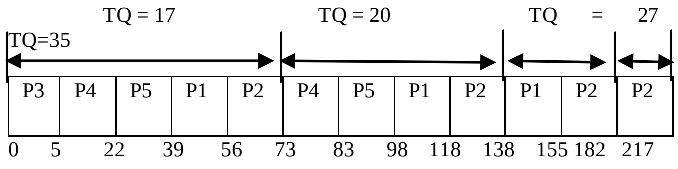

Figure. 4.6.6.3: Gantt chart for SMDRR in Table 4.6.6

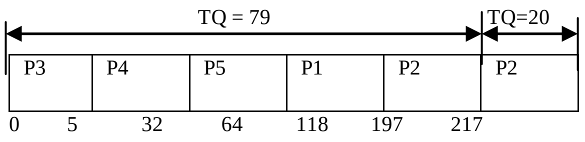

Figure. 4.6.6.4: Gantt chart for MMDRR in Table 4.6.6

CASE 2: With Arrival Time

Increasing Order:

Five process P1, P2, P3, P4, P5 arriving at time 0, 2, 5, 7, 9 respectively with burst time 10, 22, 48, 70, 74 respectively of each process shown in table 4.6.7 have been considered. Table 4.6.8 shows the comparing result of RR, SRBRR, SMDRR and our proposed algorithm (MMDRR).

Table 4.6.7 Data in increasing order with arrival time

Processes	Arrival Time	Burst Time
P1	0	10
P2	2	22
P3	5	48
P4	7	70
P5	9	74

Table 4.6.8 Comparison among RR, SRBRR, SMDRR and MMDRR

Algorithms	QT	AWT	ATAT	CS
RR	25	69.8	114.6	9
SRBRR	10, 59, 13, 2	61.6	106.4	7
SMDRR	10, 43, 11, 18, 2	77.8	122.6	10
MMDRR	10,74	49.8	94.6	4

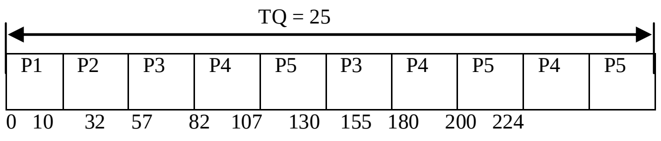

Figure. 4.6.8.1: Gantt chart for RR in Table 4.6.8

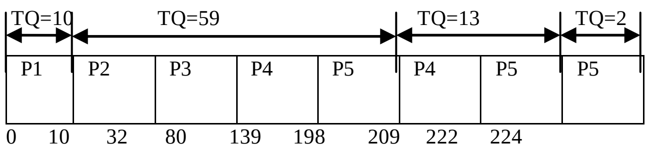

Figure. 4.6.8.2: Gantt chart for SRBRR in Table 4.6.8

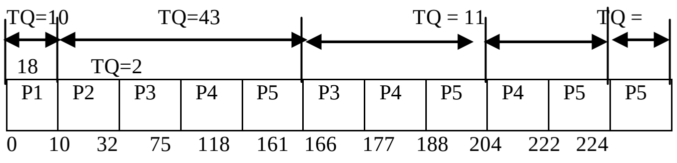

Figure. 4.6.8.3: Gantt chart for SMDRR in Table 4.6.8

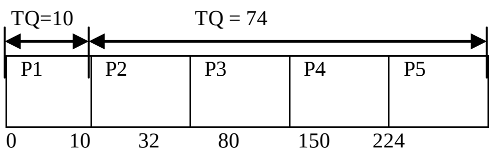

Figure. 4.6.8.4: Gantt chart for MMDRR in Table 4.6.8

Decreasing Order:

Five process P1, P2, P3, P4, P5 arriving at time 0, 6, 13, 21, 75 with burst time 73, 50, 23, 19, 5 respectively of each process shown in table 4.6.9 have been considered. Table 4.6.10 shows the comparing result of RR, SRBRR, SMDRR and our proposed algorithm (MMDRR).

Table 4.6.9 Data in decreasing order with arrival time

Processes	Arrival Time	Burst Time
P1	0	73
P2	6	50
P3	13	23
P4	21	19
P5	75	5

Table 4.6.10 Comparison among RR, SRBRR, SMDRR and MMDRR

Algorithms	QT	AWT	ATAT	CS
RR	25	67.8	101.8	7
SRBRR	73, 23, 23, 27	53.4	87.4	5
SMDRR	73, 23, 10, 27	71	105	8
MMDRR	73, 42, 8	53.4	87.4	5

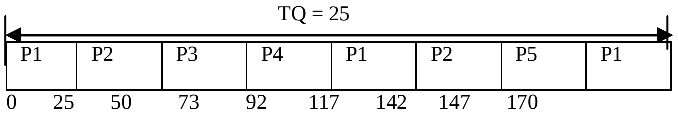

Figure. 4.6.10.1: Gantt chart for RR in Table 4.6.10

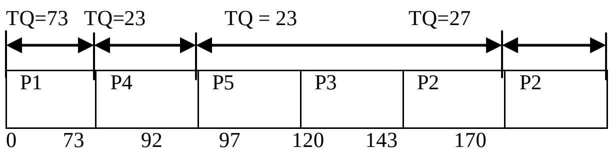

Figure. 4.6.10.2: Gantt chart for SRBRR in Table 4.6.10

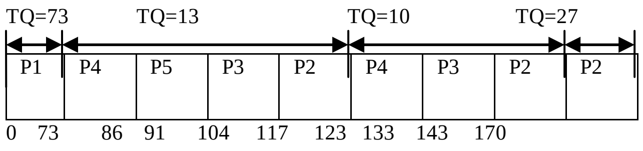

Figure. 4.6.10.3: Gantt chart for SMDRR in Table 4.6.10

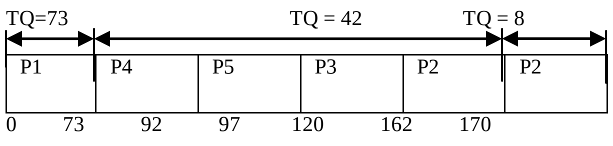

Figure. 4.6.10.4: Gantt chart for MMDRR in Table 4.6.10

Random Order:

Five process P1, P2, P3, P4, P5 arriving at time 0, 6, 8, 9, 10 with burst time 7, 15, 90, 42, 8 respectively of each process shown in table 4.6.11 have been considered. Table 4.6.12 shows the comparing result of RR, SRBRR, SMDRR and our proposed algorithm (MMDRR).

Table 4.6.11 Data in random order with arrival time

Processes	Arrival Time	Burst Time
P1	0	7
P2	6	15
P3	8	90
P4	9	42
P5	10	8

Table 4.6.12 Comparison among RR, SRBRR, SMDRR and MMDRR

Algorithms	QT	AWT	ATAT	CS
RR	25	39.6	72	8
SRBRR	7, 15, 42, 48	19.6	52	5
SMDRR	7, 18, 36, 36	23.3	55.6	7
MMDRR	7, 76, 14	19.6	52	5

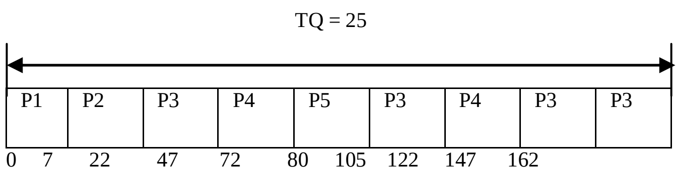

Figure. 4.6.12.1: Gantt chart for RR in Table 4.6.12

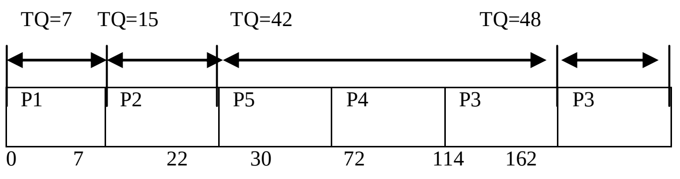

Figure. 4.6.12.2: Gantt chart for SRBRR in Table 4.6.12

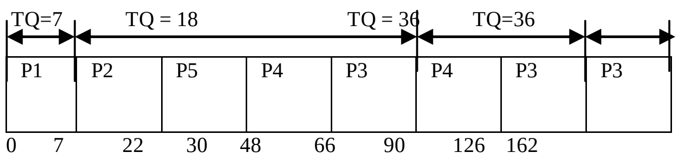

Figure. 4.6.12.3: Gantt chart for SMDRR in Table 4.6.12

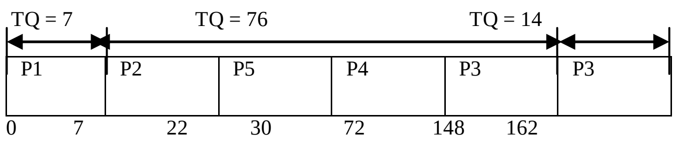

Figure. 4.6.12.4: Gantt chart for SMDRR in Table 4.6.12

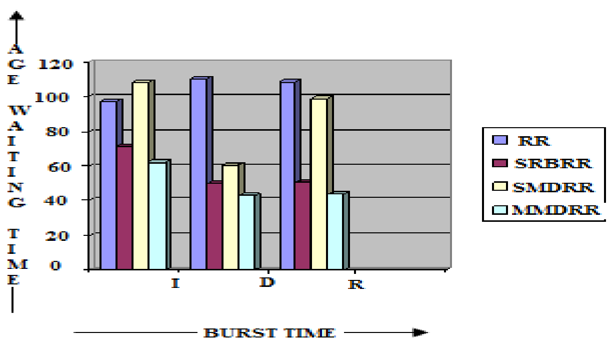

Figure 4.6.13 Avg. waiting time (RR vs. SRBRR vs. SMDRR vs. MMDRR) with arrival time= 0

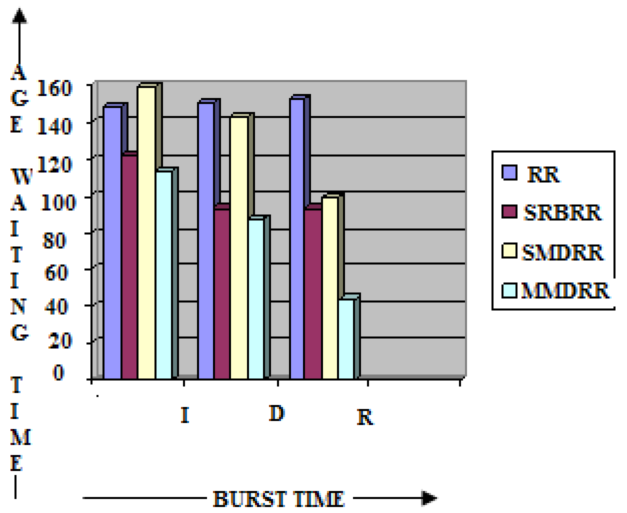

Figure 4.6.14 Avg. turnaround time (RR vs. SRBRR vs. SMDRR vs. MMDRR) with arrival time= 0

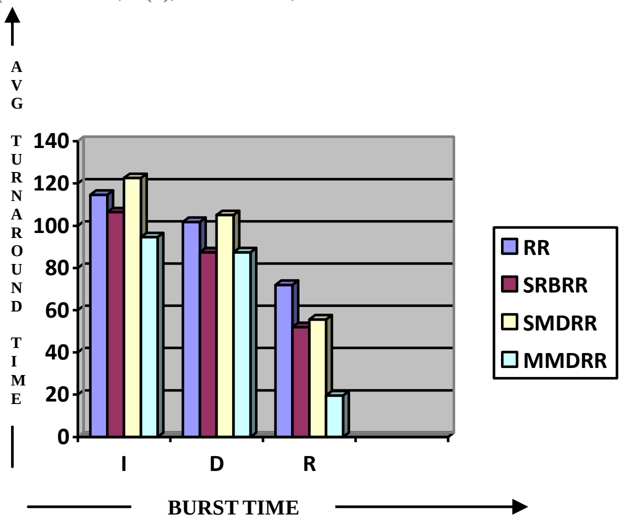

Figure: 4.6.17 Avg. turnaround time (RR vs. SRBRR vs. SMDRR vs. MMDRR) with arrival time

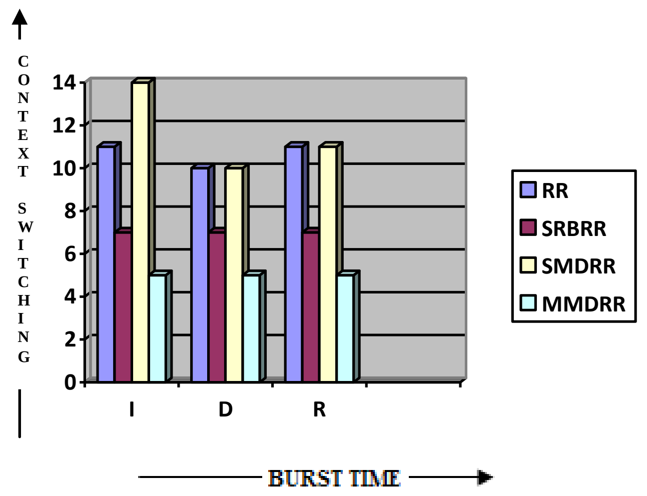

Figure 4.6.15 Context switching (RR vs. SRBRR vs. SMDRR vs. MMDRR) with arrival time= 0

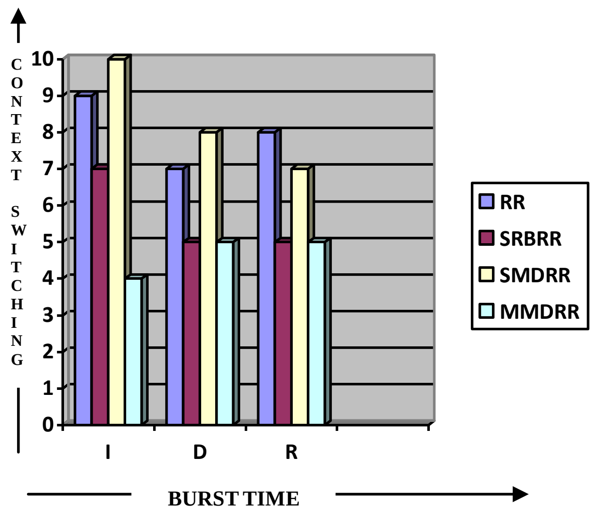

Figure: 4.6.18 Context switching (RR vs. SRBRR vs. SMDRR vs. MMDRR) with arrival time

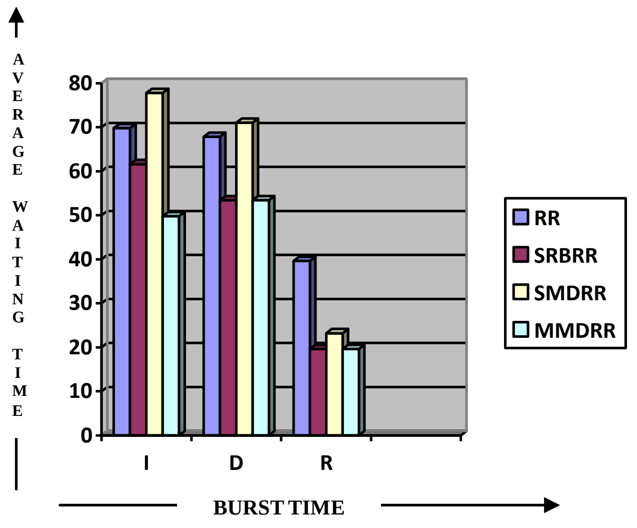

Figure 4.6.16 Avg. waiting time (RR vs. SRBRR vs. SMDRR vs. MMDRR) with arrival time

CONCLUSION

From the above experiments, MMDRR algorithm shows better results than RR algorithm, SRBRR algorithm and SMDRR algorithm in enhancing the CPU performance and its efficiency. By using our algorithm we are getting better, Average Waiting Time, Average Turnaround Time and Context Switch. As we have taken the ideal cases in calculating the TAT, WT and CS .In future we can implement this algorithm in real time operating systems.

REFERENCES

[1]. Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, “Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm”, In Proceedings of International Symposium on Computer Engineering & Technology (ISCET), Vol 17, 2010

[2]. Sourav Kumar Bhoi, Sanjaya Kumar Panda, Debashee Tarai, “Enhancing cpu performance using subcontrary mean

dynamic round robin (smdrr) scheduling algorithm" ‚JGRCS, Volume 2, No. 12, December 2011, pp.17-21

[3]. R. J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the now Running Processes", American Journal of Applied Sciences 6 (10), 2009, ISSN 1546-9239, pp. 1831-1837.

[4]. A. Silberschatz, P. B. Galvin and G. Gagne, "Operating System Principles", 7th Edn., John Wiley and Sons, 2008, ISBN 978-81-265-0962-1.

[5]. Rami Abielmona, *Scheduling Algorithmic Research,* Department of Electrical and Computer Engineering Ottawa-Carleton Institute, 2000.

[6]. R. Mohanty, H. S. Behera, K. Patwari, M. Dash, M. L. Prasanna, "Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems", IJACSA, Vol. 2, No. 2, Feb 2011, pp. 46-50.

Short Bio Data for the Author

Dr. H.S.Behera is currently working as a faculty in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India. His areas of interest include Distributed Systems, Data Mining and Soft Computing.

Sreelipa Curtis is a Final Year B.Tech student in Dept. of Computer Science & Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India.

Bijayalaxmi Panda is a Final Year B.Tech student in Dept. of Computer Science & Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India.