Volume 4, No. 3, March 2013
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

A FINEST TIME QUANTUM FOR IMPROVING SHORTEST REMAINING BURST ROUND ROBIN (SRBRR) ALGORITHM

P.Surendra Varma
Department of computer science & Engineering NRI Institute of technology
Vijayawada, Andhra Pradesh, India
Surendravarma008@gmail.com

Abstract- Round Robin (RR) performs optimally in timeshared systems because each process is given an equal amount of static time quantum. But the effectiveness of RR algorithm solely depends upon the choice of time quantum. I have made a comprehensive study and analysis of RR algorithm and SRBRR algorithm. I have proposed an improved version of SRBRR (Shortest Remaining Burst Round Robin) by assigning the processor to processes with shortest remaining burst in round robin manner using the best possible time quantum. Time quantum is computed with the help of median and highest burst time. My experimental analysis shows that ISRBRR performs better than RR algorithm and SRBRR in terms of reducing the number of context switches, average waiting time and average turnaround time.

Keywords: Operating System, Scheduling Algorithm, Round Robin, Context switch, Waiting time, Turnaround time.

INTRODUCTION

A process is an instance of a computer program that is being executed. It includes the current values of the program counter, registers, and variables. The subtle difference between a process and a program is that the program is a group of instructions whereas the process is the activity. The processes waiting to be assigned to a processor are put in a queue called *ready queue.* The time for which a process holds the CPU is known as *burst time*. *Arrival Time* is the time at which a process arrives at the ready queue. The interval from the time of submission of a process to the time of completion is the turnaround time.. *Waiting time* is the amount of time a process has been waiting in the ready queue. The number of times CPU switches from one process to another is known as *context switch*. The optimal scheduling algorithm will have minimum waiting time, minimum turnaround time and minimum number of context switches.

PRELIMINARIES

Basic Scheduling Algorithms:

First Come First Serve (FCFS):

In this algorithm, the process to be selected is *the process which requests the processor first*. This is the process whose PCB is at the head of the ready queue. Contrary to its simplicity, its performance may often be poor compared to other algorithms. FCFS may cause processes with short processor bursts to wait for a long time. If one process with a long processor burst gets the processor, all the others will wait for it to release it and the ready queue will be filled very much. This is called the *convoy effect*.

Shortest Job First (SJF):

In this strategy the scheduler arranges processes with the Burst times in the ready queue, so that the process with low burst time is scheduled first. If two processes having same burst time and arrival time, then FCFS procedure is followed.

Shortest Remaining Time First (SRTF):

This is same as the SJF with pre emption, which small modification. For scheduling the jobs system need to consider the remaining burst time of the job which is presently executed by the CPU also along with the burst time of the jobs present in the ready queue.

Priority Scheduling Algorithm:

It provides the priority to each process and selects the highest priority process from the ready queue. A priority scheduling algorithm can leave some low-priority processes in the ready queue indefinitely. If the system is heavily loaded, it is a great probability that there is a higher priority process to grab the processor. This is called the starvation problem. One solution for the starvation problem might be to gradually increase the priority of processes that stay in the system for a long time.

Round robin Scheduling Algorithm:

Round Robin (RR) is one of the oldest, simplest, and fairest and most widely used scheduling algorithms, designed especially for time-sharing systems. Here every process has equal priority and is given a time quantum after which the process is preempted. The OS using RRS, takes the first process from the ready queue, sets a timer to interrupt after one time quantum and gives the processor to that process. If the process has a processor burst time smaller than the time quantum, then it releases the processor voluntarily, either by terminating or by issuing an I/O request. The OS then proceed with the next process in the ready queue. On the other hand, if the process has a processor burst time greater than the time quantum, then the timer will go off after one time quantum expires, and it interrupts (preempts) the current process and puts its PCB to the end of the ready queue.

Any CPU scheduling algorithm relies on the following criteria. They are:

a. ***Processor Utilization:*** The ratio of busy time of the processor to the total time passes for

processes to finish. We would like to keep the processor as busy as possible.
Processor Utilization = (Processor buy time) / (Processor busy time + Processor idle time)

b. Throughput: The measure of work done in a unit time interval.

Throughput = (Number of processes completed) / (Time Unit)

c. Turnaround Time (tat): The sum of time spent waiting to get into the ready queue,

Execution time and I/O time.
tat = t(process completed) – t(process submitted)

d. Waiting Time (wt): Time spent in ready queue. Processor scheduling algorithms only affect the time spent waiting in the ready queue. So, considering only waiting time instead of turnaround time is generally sufficient.

e. Response Time (rt): The amount of time it takes to start responding to a request. This criterion is important for interactive systems.

rt = t(first response) – t(submission of request)

We, normally, want to maximize the processor utilization and throughput, and minimize tat, wt, and rt. However, sometimes other combinations may be required depending on to processes.

RELATED WORK

In the last few years different approaches are used to increase the performance of Round Robin scheduling like Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice[1], Multi-Dynamic time Quantum Round Robin (MDTQRR)[5].Min-Max Round Robin (MMRR)[2], Self-Adjustment Time Quantum in Round Robin (SARR)[10], Dynamic Quantum with Re-adjusted Round Robin (DQRRR)[11],Average Max Round Robin Algorithm (AMRR)[8]. In this paper also Efforts have been made to modify SRBRR in order to give better turnaround time, average waiting time and minimize context switches.

PROPOSED ALGORITHM

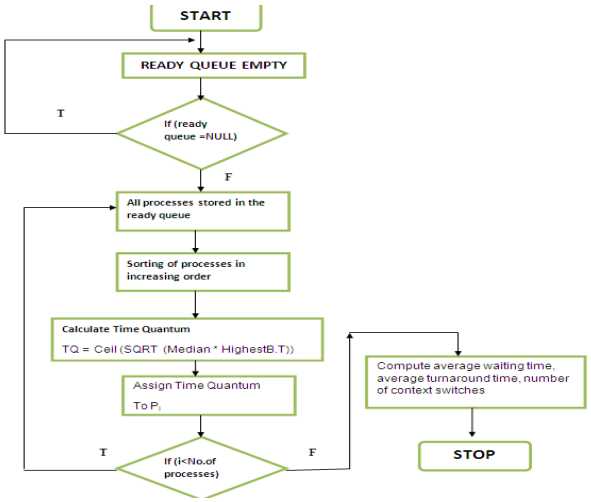
The proposed algorithm works as follows:

a. All the processes present in ready queue are sorted in ascending order.
b. While (ready queue! = NULL)
TQ = Ceil (sqrt (median * highest Burst time))
c. Assign TQ to process
Pi ->TQ
d. If (i<n) then go to step 3
e. If a new process is arrived,

Update the counter n and go to step1
End of while

f. Average waiting time, average turnaround time and Number of context switches are calculated.
g. End

Flowchart:



EXPERIMENTS & RESULTS

Assumptions:

All experiments are assumed to be performed in uniprocessor environment and all the processes are independent from each other. Attributes like burst time and priority are known prior to submission of process. All processes are CPU bound. No process is I/O bound. Processes with same arrival time are scheduled.

Illustration and Results:

Case-I :
Let us assume five processes, with increasing burst time (P1 = 13, P2 = 35, P 3 = 46, P4 = 63, p5= 97) as shown in TABLE.

Process	Burst Time
P1	13
P2	35
P3	46
P4	63
P5	97

Now, as per the algorithm Time Quantum is calculated as follows

TQ = Ceil (sqrt (median * highest Burst time))

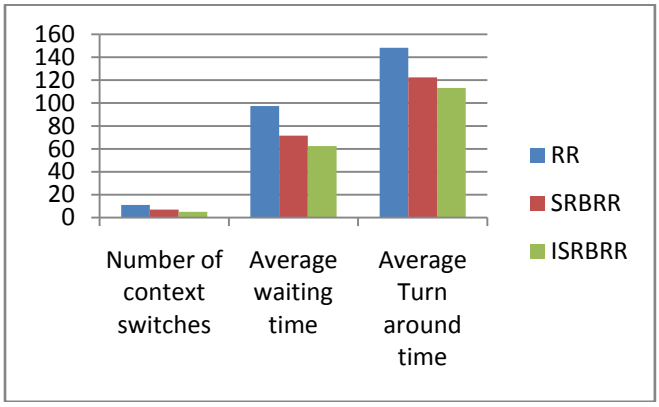
TQ = Ceil (sqrt(46 * 97)) = 67

P1	P2	P3	P4	P5	P5

0 13 48 94 157 224 254
Number of Context Switches = 5
Average Waiting Time = (0+13+48+94+157) / 5 = 62.4
Average Turn around Time = (13+48+94+157+254) / 5 = 113.2

Table I: Comparison between RR, SRBRR and Proposed algorithm (case – I)

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	25	148.2	97.4	11
SRBRR	46	122.4	71.6	7
ISRBRR	72	113.2	62.4	5



Case II:

Let us assume five processes arriving at time = 0, with decreasing burst time (P1 = 86, P2 =53, P 3 = 32, P4= 21, p5= 9) as shown in TABLE

Process	Burst Time
P1	86
P2	53
P3	32
P4	21
P5	9

Now , TQ can be calculated as follows :
TQ = Ceil (sqrt (median * highest Burst time))
TQ = Ceil (sqrt(32 * 86)) = 53

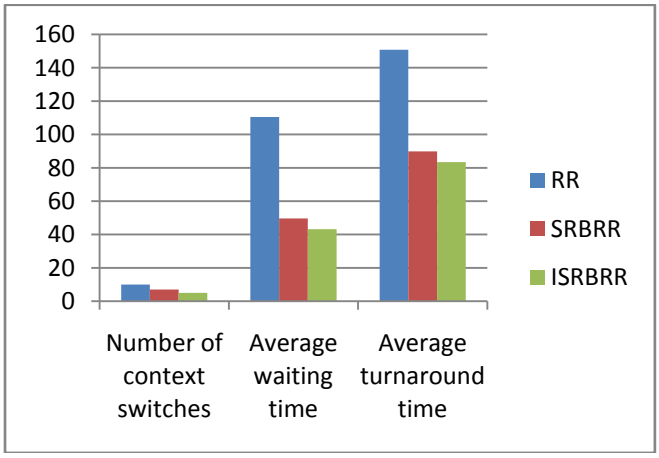
P5	P4	P3	P2	P1	P1

0 9 30 62 115 168 201

Number of Context Switches = 5
Average Waiting Time = (0+9+30+62+115) / 5 = 43.2
Average Turnaround Time = (9+30+62+115+201) / 5 = 83.4

Table 2: Comparison between RR, SRBRR and Proposed algorithm (case – II)

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	25	150.8	110.5	10
SRBRR	32	89.8	49.6	7
ISRBRR	59	83.4	43.2	5



Case-III:

Let us Assume five processes arriving at time = 0, with random burst time (P1 = 54, P2 = 99, P 3 = 5, P 4 = 27, p5= 32) as shown in TABLE

Process	Burst Time
P1	54
P2	99
P3	5
P4	27
P5	32

Now , TQ can be calculated as follows :
TQ = Ceil (sqrt (median * highest Burst time))
TQ = Ceil(sqrt(32*99)) = 57

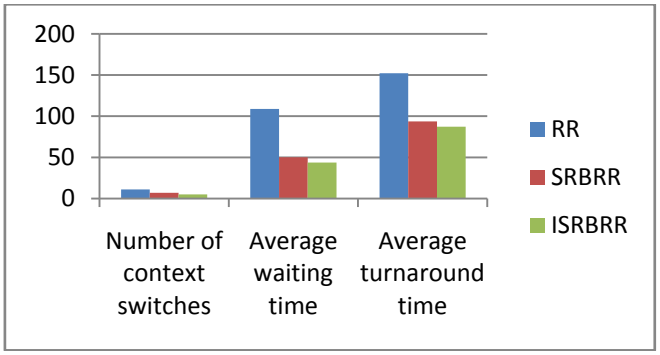
P3	P4	P5	P1	P2	P2

0 5 32 64 118 175 217

Number of Context Switches = 5
Average Waiting Time = (0+5+32+64+118) / 5 = 43.8
Average Turnaround Time = (5+32+64+118+217) / 5 = 87.2

Table III: Comparison between RR, SRBRR and Proposed algorithm (case – III)

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	25	152.2	108.8	11
SRBRR	32	93.6	50.2	7
ISRBRR	66	87.2	43.8	5



Implementation:

The algorithm is implemented using C language and its code is as follows:

Source Code

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int st[10];
int get_tq(int b[],int s)
{
int i,j,maxbt,tmp,hbt,median;
float k,l,m;
for(i=0;i<s;i++)
{
for(j=i+1;j<s;j++)
{
 if (b[i]>b[j])
```

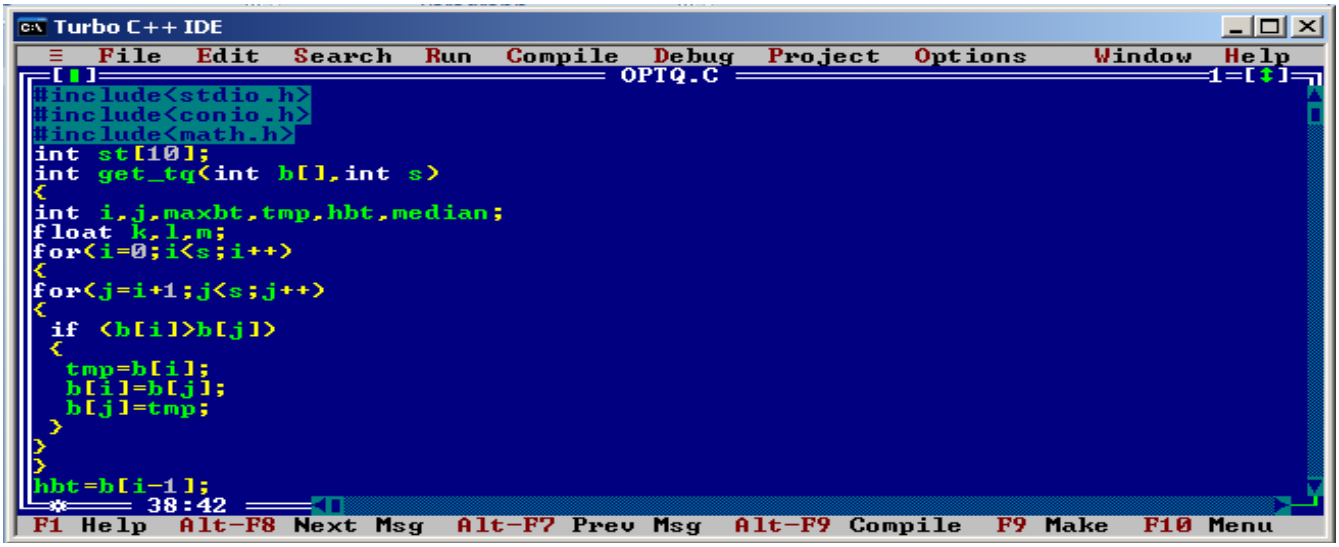
```
 {
  tmp=b[i];
  b[i]=b[j];
  b[j]=tmp;
 }
}
}
hbt=b[i-1];
median=b[i/2];
for(i=0;i<s;i++)
st[i]=b[i];
l=(float)hbt;
m=(float)median;
k=sqrt((l*m));
return(ceil(k));
}
void main()
{
int bt[10],wt[10],tat[10],n,tq;
int i,count=0,swt=0,stat=0,temp,sq=0;
float awt=0.0,atat=0.0;
clrscr();
printf("Enter number of processes:");
scanf("%d",&n);
printf("Enter burst time for sequences:");
for(i=0;i<n;i++)
{
scanf("%d",&bt[i]);
st[i]=bt[i];
}
tq=get_tq(st,n);
printf("\ntime quantum is computed by ceil((highestbt+Median)/2) = %d\n",tq);
while(1)
{
for(i=0,count=0;i<n;i++)
{
temp=tq;
if(st[i]==0)
{
count++;
continue;
}
if(st[i]>tq)
st[i]=st[i]-tq;
else
if(st[i]>=0)
{
temp=st[i];
st[i]=0;
}
sq=sq+temp;
tat[i]=sq;
}
if(n==count)
break;
}
for(i=0;i<n;i++)
{
wt[i]=tat[i]-bt[i];
swt=swt+wt[i];
stat=stat+tat[i];
}
awt=(float)swt/n;
atat=(float)stat/n;
//printf("Process_no\t Burst time\t Wait time\t Turn around time\t");
//for(i=0;i<n;i++)
//printf("%d\t %d\t %d\t %d\t",i+1,bt[i],wt[i],tat[i]);
printf("\nAvg waiting time is %f\nAvg turn around time is %f",awt,atat);
getch();
}
```

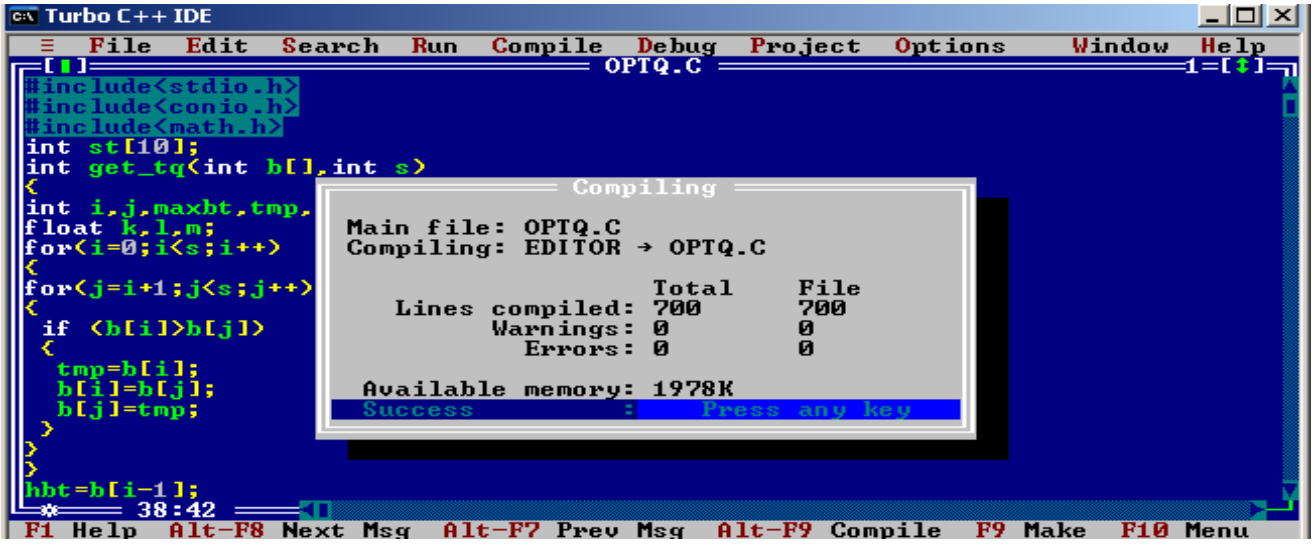
Output:

Enter number of processes:5
Enter burst time for sequences:
13
35
46
63
97
Time quantum is computed by ceil(Sqrt(highestbt*Median)) = 67
Avg waiting time is 62.400002
Avg turn around time is 113.199997

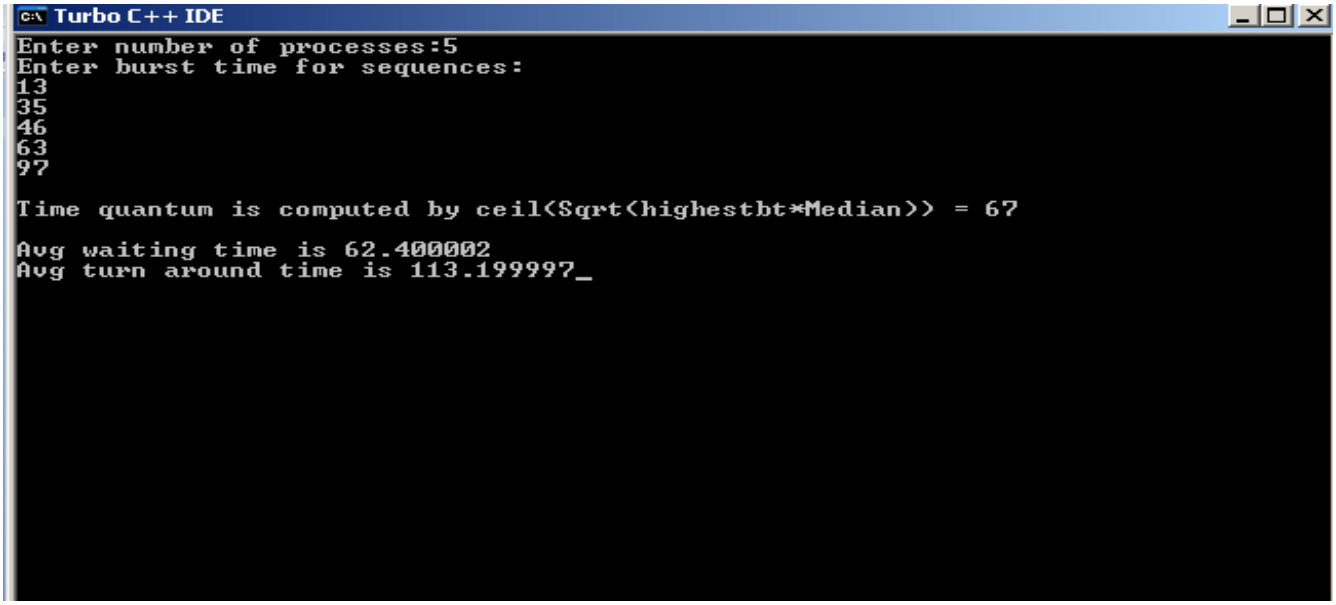
Simulation and Screen shots

Turbo C++ is used in order to simulate the source code. Here are some screen shots of simulation process.





Output simulation for case-I



CONCLUSION AND FUTURE WORK

From the above comparisons i can conclude that the proposed algorithm is performing better than the static RR algorithm and SRBRR algorithm in terms of average waiting time, average turnaround time and number of context switches. In future work, processes at different arrival times can be considered for the proposed algorithm.

REFERENCES

[1]. Sarojhiranwal and D.r. K.C.Roy"Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice".volume 2,issue 3.

[2]. Sanjay Kumar Panda and Saurav Kumar Bhoi, "An Effective Round Robin Algorithm using Min-Max Dispersion Measure" ISSN : 0975-3397 ,Vol. 4 No. 01, January 2012.

[3]. "Tanebaun, A.S., 2008" Modern Operating Systems. 3rd Edn., Prentice Hall, ISBN: 13:9780136006633, pp: 1104.

[4]. "Silberschatz, A., P.B. Galvin and G. Gagne, 2008" Operating Systems Concepts. 7th Edn., John Wiley and Sons, USA., ISBN: 13: 978-0471694663, pp: 944.

[5]. H. S. Behera, Rakesh Mohanty, Sabyasachi Sahu and Sourav Kumar Bhoi." Comparative performance analysis of multi-dynamic time quantum round robin (mdtqrr) algorithm with arrival time", ISSN : 0976-5166, Vol. 2, No. 2, Apr-May 2011.

[6]. "Tarek Helmy, Abdelkader Dekdouk" Burst Round Robin: As a Proportional-Share Scheduling Algorithm, IEEE Proceedings of the fourth IEEE-GCC Conference on towards Techno-Industrial Innovations, pp. 424-428, 11-14 November,2007

[7]. "Yaashuwanth .C & R. Ramesh" Inteligent time slice for round robin in real time operating system, IJRRAS 2 (2), February 2010.

[8]. Pallab banerjee, probal banerjee, shweta sonali dhal,"Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling

Algorithm using static Time Quantum",IJITEE,ISSN: 2278-3075, Volume-1, Issue-3, August 2012.

[9]. J. Nieh, C. Vaill and H. Zhong, "Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler", Proceedings of the USENIX

[10]. R. J. Matarneh, "Seif-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Proceses", American Journal of Applied Sciences 6 (10), pp. 1831-1837, 2009.

[11]. H. S. Behera, R. Mohanty, and D. Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis," vol. 5, no. 5, pp. 10-15, August 2010.

[12]. A. Bhunia, "Enhancing the Performance of Feedback Scheduling", IJCA, vol. 18, no. 4, pp. 11-16, March 2011.

[13]. "Prof. Rakesh Mohanty, Prof. H. S. Behera, Khusbu Patwari, Manas Ranjan Das, Monisha Dash,Sudhashree" Design and Performance Evaluation of a New Proposed Shortest Remaining Burst RoundRobin (SRBRR) Scheduling Algorithm, Am. J. Applied Sci., 6 (10): 1831-1837, 2009.

Short Bio data for the Author profile

P.Surendra Varma received his M.Tech Computer science Engineering from Acharya Nagarjuna university campus. He is working as an Associate Professor in NRI institute of technology, Vijayawada. His research interests includes Bioinformatics, compression techniques, operating systems, theory of computation, compiler design, programming languages, data mining and warehousing, software engineering.