ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 8, October 2012

Improved Shortest Remaining Burst Round Robin (ISRBRR) Using RMS as its time quantum

P.Surendra Varma

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 RMS as its time quantum. Time quantum is computed as the root mean square of the burst times. 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.***

Index Terms— Operating System, Scheduling Algorithm, Round Robin, Context switch, Waiting time, Turnaround time,RMS

I. Introduction

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

A) CPU utilization

We want to keep the CPU as busy as possible that means CPU is not free during the execution of processes. Conceptually the CPU utilization can range from 0 to 100 percent.

(B) Response time

Response time is the time from the submission of a request until the first response is produced.

(C) Throughput:

One measure work is the number of processes that are completed per time unit that means the number of tasks per second which the scheduler manages to complete the tasks.

(D) Turnaround Time:

The time interval from the time of submission of a process to the time of completion is the turnaround time. Total turnaround time is calculation is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

(E) Waiting Time: The waiting time is not the measurement of time when a process executes or does I/O completion; it affects only the amount of time of submission of a process spends waiting in the ready queue. We keep average waiting time should be less.

Manuscript received Sep 15, 2012.

P.Surendra Varma, *Department of computer science & Engineering, NRI Institute of technology, Vijayawada ,India.*

.

.

II. Preliminaries

A **process** is an instance of a computer program that is being executed. 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.

Basic Scheduling Algorithms

First-Come First-Served (FCFS) is a nonpreemptive algorithm that assigns the CPU to the process in the ready queue that has been waiting for the longest time. This is a simple algorithm and it is not used very often in modern operating systems. A long process could cause a delay for all other processes that arrive after that process.

Shortest Process (Job) Next (SJN) is another nonpreemptive algorithm that attempts to decrease the average waiting time (and response time) of the system. This algorithm performs better than FCFS, however, it is not fair to long processes. In general preemptive scheduling algorithms are preferred due to their abilities to switch the CPU to another process even when the current running process is not completed.

In***Round Robin*** **(RR)** scheduling a time slice is defined and the CPU is assigned to a process for a maximum of one time slice or until the process releases the CPU (whichever comes first). This algorithm requires more overheads but it is fair to all processes and performs better than nonpreemptive scheduling algorithms.

Shortest Remaining Time Next **(SRTN)** is a preemptive version of SJN algorithm where the remaining processing time is considered for assigning CPU to the next process.

Highest Response Ratio Next **(HRRN)** selects a process with the largest ratio of waiting time over service time. This guarantees that a process does not starve due to its requirements.

Feedback Queue **(FQ)** scheduling algorithm partitions the ready processes into several separate queues and the processes are assigned to one queue and they are allowed to move between queues. Each queue has its own scheduling algorithm.

III.Related Work

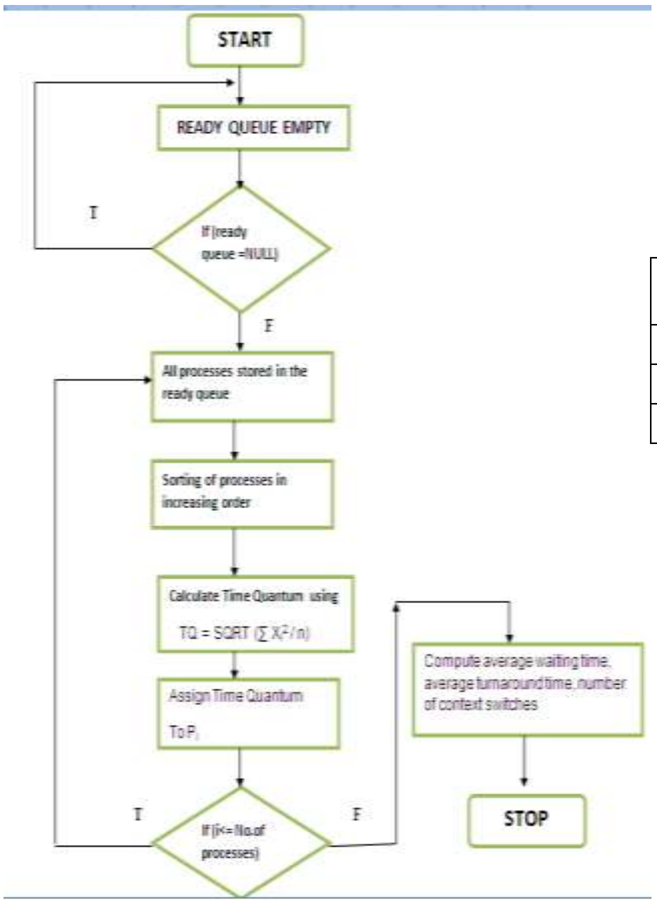
Efforts have been made to modify RR in order to give better average turnaround time, average waiting time and minimize context switches.

IV. Proposed Algorithm

A. The proposed algorithm works as follows:

1. First of all check ready queue is empty.
2. When ready queue is empty then all the processes are assigned into the ready queue.
3. All the processes are rearranged in increasing order that means smaller burst time process get higher priority and larger burst time process get lower priority.
4. While (ready queue! = NULL)
5. Calculate Time Quantum:
 $TQ = SQRT (\sum X_i^2 / n)$
6. Assign Time Quantum to each process:
 $P_i \leftarrow TQ$ and $i=i+1$
7. If (i< Number of processes) then go to step 6.
8. If a new process is arrived update the ready queue, go to step 2.
9. End of While
10. Calculate average waiting time, average turnaround time, number of context switches.
11. End

B. FLOW CHART


START
READY QUEUE EMPTY
T
If (ready queue =NULL)
F
All processes stored in the ready queue
Sorting of processes in increasing order
Calculate Time Quantum using TQ = SQRT (∑ X²/n)
Assign Time Quantum To Pi
T
If (i<=No.of processes)
F
Compute average waiting time, average turnaround time, number of context switches
STOP

Include a note with your final paper indicating that you

V. Experiments & Results

A. 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.

B. Illustration and Results

Case-I :

Let us Assume five processes, with increasing burst time (P1 = 9, P2 = 14, P3 = 21, P4 = 26, p5= 49) as shown in TABLE.

Process	Burst Time
P1	9
P2	14
P3	21
P4	26
P5	49

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

$TQ = SQRT (\sum X_i^2 / n)$

$TQ = SQRT ((9^2+14^2+21^2+26^2+49^2) / 5) = 28$

|←----------- TQ= 28----------------→| ←TQ=21→

P1	P2	P3	P4	P5	P5

0 9 23 44 70 98
119

Number of Context Switches = 5

Average Waiting Time = (0+9+23+44+70) / 5 = 29.2

Average Turnaround Time = (9+23+44+70+119) / 5 = 53

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	25	58	34.2	6
SRBRR	46	57.2	33.4	7
ISRBRR	72	53	29.2	5

Table 1: Comparison between RR, SRBRR and Proposed algorithm (case – I)

Case II :

Let us assume five processes arriving at time = 0, with decreasing burst time (P1 = 54, P2 =32, P3 = 24, P4= 18, p5= 13) as shown in TABLE

Process	Burst Time
P1	54
P2	32
P3	24
P4	18
P5	13

Now, TQ can be calculated as follows:

$TQ = SQRT (\sum X_i^2 / n)$

$TQ = SQRT ((54^2+32^2+24^2+18^2+13^2) / 5) = 32$

|←--------- TQ=32 ------------------→| ←TQ=22→|

P5	P4	P3	P2	P1	P1

0 13 31 55 87 119 141

Number of Context Switches = 5

Average Waiting Time = (0+13+31+55+87) / 5 = 37.2

Average Turnaround Time = (13+31+55+87+141) / 5 = 65.4

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	25	109.8	81.6	7
SRBRR	32	70.2	42	7
ISRBRR	59	65.4	37.2	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 :

$TQ = SQRT\ (\sum X_i^2 / n)$
$TQ = SQRT\ ((54^2+99^2+5^2+27^2+32^2) / 5) = 54$

|←---------TQ=54 -------------------→|←TQ=45→|

P3	P4	P5	P1	P2	P2

0 5 32 64 118 172 217

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

Algorithm	Time Quantum	Avg.TAT	Avg.WT	CS
RR	25	152.2	108.8	11
SRBRR	32	93.6	50.2	7
ISRBRR	66	87.8	43.8	5

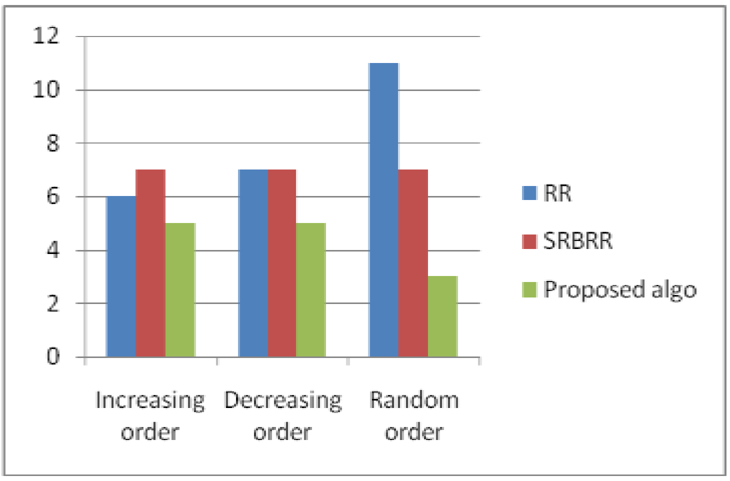


Fig (a): Comparison for number of Context Switches for each case

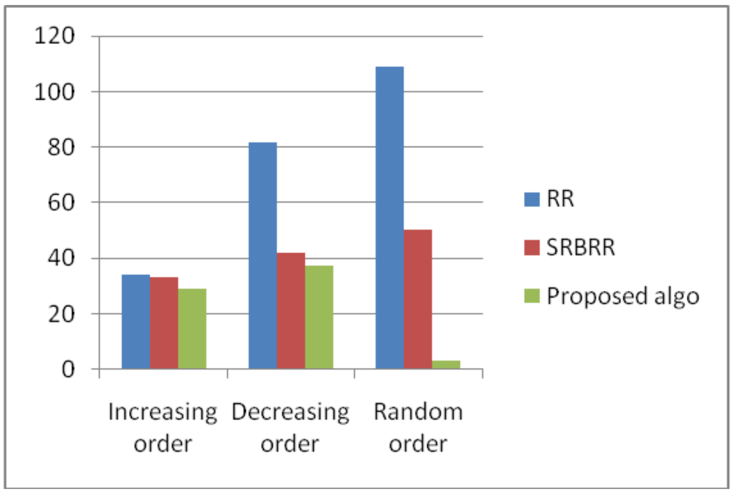


Fig (b): Comparison of Average waiting time for each case

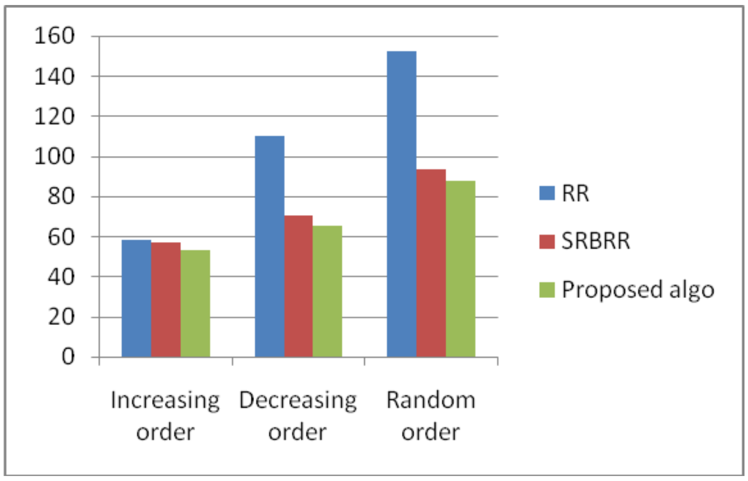


Fig (c): Comparison of Average turnaround time for each case

C. Implementation

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

Source Code

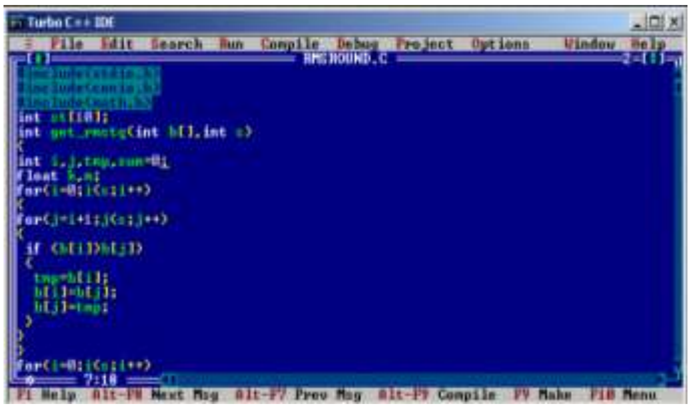
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int st[10];
int get_rmstq(int b[],int s)
{
int i,j,tmp,sum;
float k,m;
for(i=0;i<s;i++)
{
for(j=i+1;j<s;j++)
{
 if (b[i]>b[j])
 {
  tmp=b[i];
  b[i]=b[j];
  b[j]=tmp;
 }
}
}
for(i=0;i<s;i++)
{
st[i]=b[i];
sum=sum+b[i]*b[i];
}
m=sum/s;
k=sqrt(m);
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_rmstq(st,n);
printf("\ntime quantum is computed by root mean
square is = %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

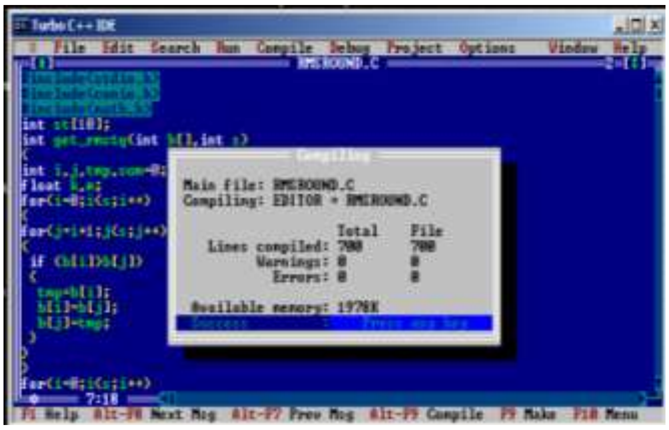
```
Enter number of processes:5
Enter burst time for sequences:9
14
21
26
49
time quantum is computed by root mean square is = 28
Avg waiting time is 29.200001
Avg turnaround time is 53.000000
```

D. Simulation and Screen shots

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


Turbo C++ IDE
File Edit Search Run Compile Debug Project Options Window Help
RMSROUND.C
7:18
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int st[10];
int get_rmstq(int b[],int s)
{
int i,j,tmp,sum=0;
float k,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;
 }
}
}
for(i=0;i<s;i++)
```

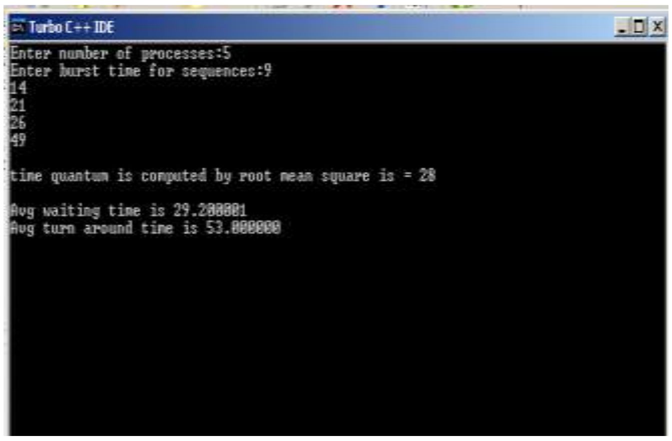

Turbo C++ IDE
File Edit Search Run Compile Debug Project Options Window Help
RMSROUND.C
Compiling
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
[illegible]
[illegible]
[illegible]
int st[10];
int get_rmsty(int b[],int s)
{
int i,j,tmp,com=0;
float k,a;
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;
 }
}
}
for(i=0;i<s;i++)
```

```
Main file: RMSROUND.C
Compiling: EDITOR → RMSROUND.C

                     Total      File
  Lines compiled:    700        700
        Warnings:    0          0
          Errors:    0          0

 Available memory: 1970K
 Success          :   Press any key
```



```
Enter number of processes:5
Enter burst time for sequences:9
14
21
26
49

time quantum is computed by root mean square is = 28

Avg waiting time is 29.200001
Avg turn around time is 53.000000
```

VI. 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] "Silberschatz, A., P.B. Galvin and G. Gagne, 2004" Operating Systems Concepts. 7th Edn., John Wiley and Sons, USA., ISBN: 13: 978-0471694663, pp: 944.

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

[3] "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 Round Robin(SRBRR) Scheduling Algorithm, Am. J. Applied Sci., 6 (10): 1831-1837, 2009.

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

[5] William Stallings, "Operating Systems: Internals and Design Principles" 6th edition, Prentice Hall, ISBN-13:978-0136006329

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