An Improved Round Robin Based Scheduling Algorithm with Dynamic Time Quantum Using Contraharmonic Mean

K.Rohini[#1], B.VijayaLakshmi[#2], M.Swapna[#3], V.Vasudha[#4]

[#]*Department of Computer Science and Engineering*
Gayatri vidya parishad college of engg. for women, Madhurawada, Visakhapatnam, Andhrapradesh, India.

[1]krohini@gvpcew.ac.in
[2]bvlakshmi@gvpcew.ac.in
[3]mswapna@gvpcew.ac.in
[4]vvasudha@gvpcew.ac.in

***Abstract*—** The main objective of this paper is to improve the Round Robin scheduling algorithm using the dynamic time slice concept. Round Robin, considered as the most widely adopted CPU scheduling algorithm, undergoes severe problems directly related to quantum size. If time quantum chosen is too large, the response time of the processes is considered too high. On the other hand, if this quantum is too small, it increases the overhead of the CPU. We have made a comprehensive study and analysis of RR algorithm, SRBRR (Shortest Remaining Burst Round Robin) algorithm and proposed a new Improved-RR version of SRBRR by assigning the processor to processes with shortest remaining burst in round robin manner using the Contraharmonic Mean as its time quantum; the idea of this approach is to make the operating systems adjusts the time quantum according to the burst time of the set of waiting processes in the ready queue. Time quantum is computed as the Contraharmonic Mean of the burst times. Our experimental analysis shows that IRRCM (Improved Round Robin using Contraharmonic Mean) 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.**

I. INTRODUCTION

Operating System is a program that controls the execution of application programs and implements an interface between the user of a computer and the computer hardware. In multitasking and multiprocessing environment the way the processes are assigned to run on the available CPUs is called *scheduling*. Main goal of the scheduling is to maximize the different performance metrics such as CPU utilization, throughput and to minimize response time, waiting time and turnaround time. The scheduling is used in the real time applications like routing of data packets in computer networking, controlling traffic in airways, roadways and railways etc. In Round Robin (RR) every process has equal priority and is given a time quantum after which the process is preempted. Although RR gives improved response time and uses shared resources efficiently, its limitations are larger waiting time, larger turnaround time for processes with variable CPU bursts due to use of static time quantum. This motivates us to implement RR algorithm with sorted remaining burst time with dynamic time quantum based on contraharmonic mean concept.

A. Preliminaries

Modern Operating Systems are moving towards multitasking environments which mainly depends on the CPU scheduling algorithm since the CPU is the most effective or essential part of the computer. The idea of multi-programming is to execute a process until it must wait, typically for the completion of some I/O request. The CPU is one of the primary computer resources. The CPU scheduling is central to operating system design. Round Robin is considered the most widely used scheduling algorithm in CPU scheduling [7,8], also used for flow passing scheduling through a network device [11].CPU Scheduling is an essential operating system task, which is the process of allocating the CPU to a specific process for a time slice. Scheduling requires careful attention to ensure fairness and avoid process starvation in the CPU. This allocation is carried out by software known as scheduler and dispatcher [7, 8]. A primary function of an operating system is to determine which processes (and, in turn, users) get to utilize the CPU(s).CPU scheduling can be done at three different levels as shown in figure 1.

a. Long-term Scheduling– also known as batch scheduling. Decide which jobs/processes are allowed into the system.
b. Short-term Scheduling– or interactive scheduling. Decide from a collection of ready processes which gets the CPU next.
c. Medium-term Scheduling– or memory scheduling. Decide if/when a process should be "swapped out" or back in based on memory available.

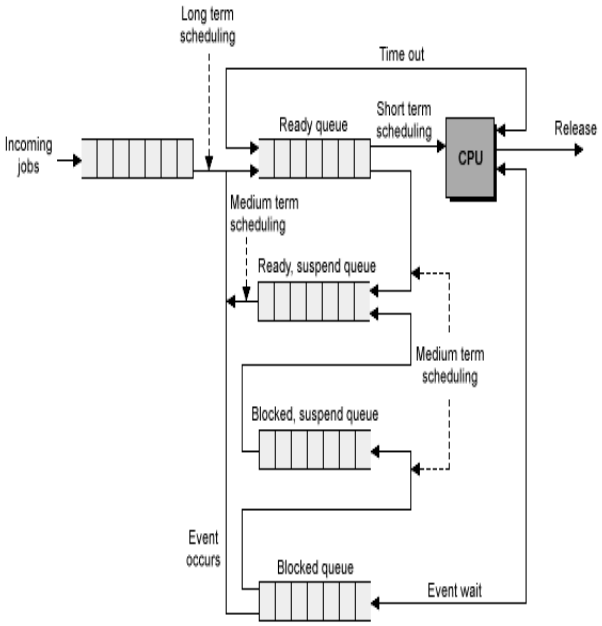


Fig. 1 Queuing diagram for scheduling

The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler [7]. A typical process alternates between the need for CPU and the need for I/O service throughout its lifespan. This is called the CPU-I/O burst cycle. It is this fact that makes multiprogramming essential. When a process needs I/O, it's good to have another process ready to move in and take advantage of the available CPU resource. The amount of time that it can make use of the CPU is known as its *CPU burst time*. 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*. The time at which a process arrives is its *arrival time*.

A program in execution is called a *process*.

Processes may be categorized as [7]:

CPU-bound– process does not need much I/O service, almost always want the CPU

I/O-bound– short CPU burst times, needs lots of I/O service

Interactive– short CPU burst times, lots of time waiting for user input (keyboard, mouse)

Moreover, we should distinguish between the two schemes of scheduling: preemptive and non preemptive algorithms. Preemptive algorithms are those where the burst time of a process being in execution is preempted when a higher priority process arrives. Non preemptive algorithms are used where the process runs to complete its burst time even a higher priority process arrives during its execution time. The type of processes in the system will affect the performance of scheduling algorithms. A short-term CPU scheduling decision is needed when a process:

i. Switches from a running to a waiting state (non-preemptive)
ii. Switches from a running to a ready state (preemptive)
iii. Switches from a waiting to a ready state (preemptive)
iv. terminates (non-preemptive)

There are many different scheduling algorithms which varies in efficiency according to the holding environments, which means what we consider a good scheduling algorithm in some cases which is not so in others, and vice versa. The Criteria for a good scheduling algorithm depends, among others, on the following measures [8]:

a. Fairness: Processes get close to equal shares of the CPU
b. Efficiency: Keep resources as busy as possible
c. Throughput: Number of processes that complete per unit time
d. Waiting Time: Time a process spends waiting in kernel's ready queue
e. Turnaround Time: Time from process start to its completion
f. Response Time: Amount of time from when a request was first submitted until first response is produced

B. Scheduling algorithms

When there are number of processes in the ready queue, the algorithm which decides the order of execution of those processes is called *scheduling algorithm* [8]. The various well known CPU scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF), Highest Response Ratio Next (HRRN) and Priority. All the above four algorithms are non-preemptive in nature and are not suitable for time sharing systems. Shortest Remaining Time Next (SRTN) and Round Robin (RR) are preemptive in nature. RR is most suitable for time sharing systems.

II. RELATED WORK

The static time quantum which is a limitation of RR was removed by taking dynamic time quantum. 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) [2], Min-Max Round Robin (MMRR) [3], Self-Adjustment Time Quantum in Round Robin (SARR) [4], Dynamic Quantum with Re-adjusted Round Robin (DQRRR) [5], Average Max Round Robin Algorithm (AMRR) [6]. In this paper efforts have been made to modify SRBRR [9] in order to give better turnaround time, average waiting time and minimize context switches.

III. PROPOSED ALGORITHM

In our IRRCM algorithm, the jobs are sorted in ascending order of their burst time to give better turnaround time and waiting time. Here Dynamic time quantum is calculated by taking Contraharmonic mean of the burst times, which generates optimal time quantum to reduce waiting time and turnaround time in this algorithm.

The Proposed algorithm works as follows

i. All the processes present in ready queue are sorted in ascending order.
ii. While (ready queue!= NULL)
TQ = round (Contraharmonic mean (burst time of all the Processes in ready queue))
iii. Assign TQ to process
$P_i \leftarrow$TQ (i=0, 1…n where n is the no. of processes)
iv. If (i<n) then go to step 3
v. If a new process is arrived, Update the n value and go to step1
End of while
vi. Average waiting time, average turnaround time and no. of context switches are calculated
vii. End

IV. 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*

Our examples consists of several input and output parameters. The input parameters consist of burst time, time quantum and the number of processes. The output parameters consist of average waiting time, average turnaround time and number of context switches.
In IRRCM the time quantum is calculated using Contraharmonic mean.
For *n* values, the contra- harmonic mean is
$(x_1^2 + x_2^2 + ... + x_n^2)/(x_1 + x_2 + ... + x_n)$

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 below TABLE.

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

TQ = round $((13^2+35^2+46^2+63^2+97^2) / (13+35+46+63+97))$
= round (66.48819)
= 66

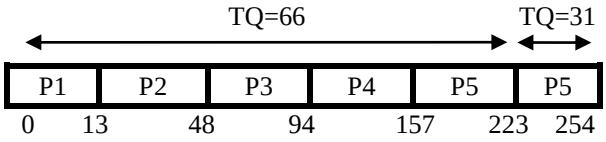
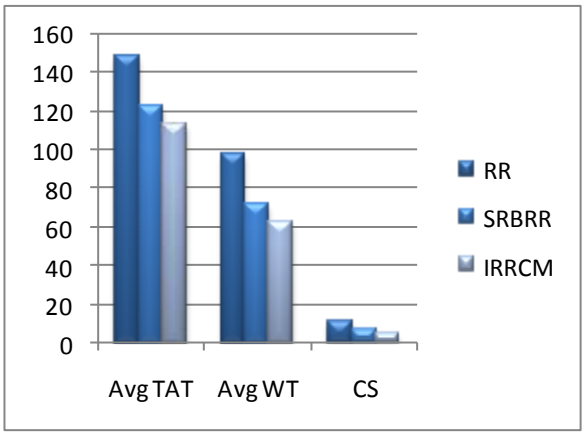


TABLE 1:
COMPARISON BETWEEN RR, SRBRR AND PROPOSED ALGORITHM (CASE – I)

Algorithm	Time Quantum	Avg TAT	Avg WT	CS
RR	25	148.2	97.4	11
SRBRR	46,34,17	122.4	71.6	7
IRRCM	**66,31**	**113.2**	**62.4**	**5**



Case-II:
Let us assume five processes, with increasing burst time (P1 = 54, P2 =32, P3 = 24, P4= 18, p5=13) as shown in below TABLE.

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

TQ = round $((54^2+32^2+24^2+18^2+13^2) / (54+32+24+18+13))$
= round (35.52482)
= 36

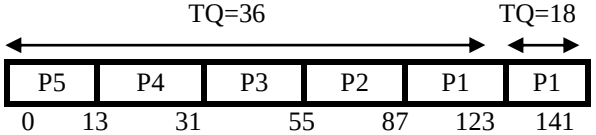
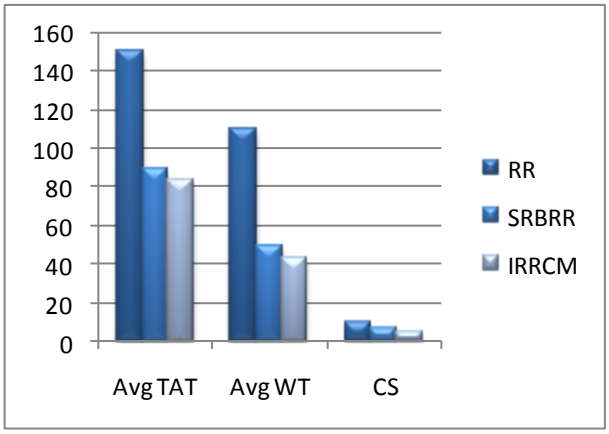


TABLE 2:
COMPARISON BETWEEN RR, SRBRR AND PROPOSED ALGORITHM (CASE – II)

Algorithm	Time Quantum	Avg TAT	Avg WT	CS
RR	25	150.8	110.5	10
SRBRR	32	89.8	49.6	7
IRRCM	**36,18**	**83.4**	**43.2**	**5**



Case-III:
Let us assume five processes, with increasing burst time (P1 = 54, P2 = 99, P 3 = 5, P 4 = 27, p5= 32) as shown in below TABLE.

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

TQ = round $((54^2+99^2+5^2+27^2+32^2)$ / (54+99+5+27+32))
= round (66.79724)
= 67

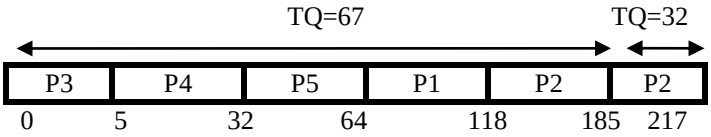
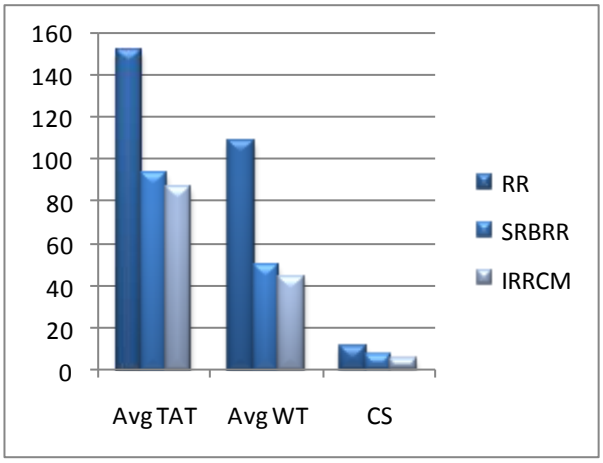


TABLE 3:
COMPARISON BETWEEN RR, SRBRR AND PROPOSED ALGORITHM (CASE – III)

Algorithm	Time Quantum	Avg TAT	Avg WT	CS
RR	25	152.2	108.8	11
SRBRR	32,45,22	93.6	50.2	7
IRRCM	**67,32**	**87.2**	**43.8**	**5**



V. CONCLUSION AND FUTURE WORK

A comparative study of simple RR algorithm and proposed one is made. It is concluded that our new proposed algorithm (IRRCM) 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 thereby reducing the overhead and saving of memory space. 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.a

[2]. 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.

[3]. 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.

[4]. R. J. Matarneh, "Self-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.

[5]. 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.

[6]. 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. F

[7] Silberschatz, Galvin and Gagne, Operating systems concepts,8th edition, Wiley, 2009.

[8] Lingyun Yang, Jennifer M. Schopf and Ian Foster,"Conservative Scheduling: Using predictive variance to improve scheduling decisions in Dynamic Environments",SuperComputing 2003, November 15-21, Phoenix, AZ, USA.

[9]. "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.

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

[11] Weiming Tong, Jing Zhao, "Quantum Varying Deficit Round Robin Scheduling Over Priority Queues", International Conference on Computational Intelligence and Security. Pp.252- 256, China, 2007.

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

