JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-3, Issue-11 E-ISSN: 2347-2693

Min-Max Simulation: An Execution Design of Process Scheduling Algorithm

Karan Sukhija

Research Scholar, DCSA, PU, Chandigarh
rs.karan.sukhija@gmail.com

www.ijcseonline.org

Received: Oct/23/2015 Revised: Nov/06/2015 Accepted: Nov/18/2015 Published: Nov/27/2015

Abstract— Job scheduling is an elementary characteristic of an operating system. The perception is to have system resources shared by a number of processes. A number of steps need to be performed to execute a program. Instructions and data must be loaded into main memory, I/O devices and files must be initialized, and other resources must be prepared. The efficiency of a system solely is subject to the use of job scheduling algorithm in a multi-programmed system. This paper begins with a brief representation of task or job sets, followed by a discussion about different type of job scheduling algorithms. In addition, the elaboration of comparative study of the entire scheduling algorithm along with proposed work is also given. This manuscript represents the simulation design of proposed CPU scheduling algorithm called MIN-MAX which is both preemptive and non-preemptive in nature. This work encompasses a software tool which produces a wide-ranging simulation of a number of CPU scheduling algorithms and provides the output in the form of scheduling performance metrics. The main objective of the paper is to analyze the performance of different algorithms with the proposed algorithm that results in minimum average waiting time and context switches. The major focus is to improve the system efficiency in multi programming system and also reduces the starvation problem among minimum and maximum burst time processes.

Keywords— Process Scheduling, First Come First Serve (FCFS), Round Robin (RR), MIN-MAX Algorithm, Simulation Design, Starvation, Complexity Analysis.

I. INTRODUCTION

Maximum system utilization is obtained with multiprogramming concept. For this purpose, several processes are kept in memory at one time and every time a running process has to wait, until another process can take over use of the CPU. Scheduling of the CPU is fundamental to operating system design. Process execution consists of a cycle of a CPU time burst and an I/O time burst. Processes switches between these two states (i.e. CPU burst and I/O burst). Eventually, the final CPU burst ends with a system request to terminate execution. Dispatcher module gives control of the CPU to the process selected by the short term scheduler that involves switching context, switching to user mode, jumping to the proper location in the user program to restart that program. A process migrates between various scheduling queues during its lifetime as described in below mentioned table 1 [1].

Queues	Description
Job queue	Set of all processes in the system
Ready queue	Set of all processes residing in main memory, ready and waiting to execute
Device queue	Set of processes waiting for an I/O device

Table 1: Process Scheduling Queues

Process is an executing instance of a program which requires a set of resources that are allocated by the CPU. As a process executes, it changes state among five basic states as depicts in Fig.1 namely new, ready, running, waiting and terminated [1].

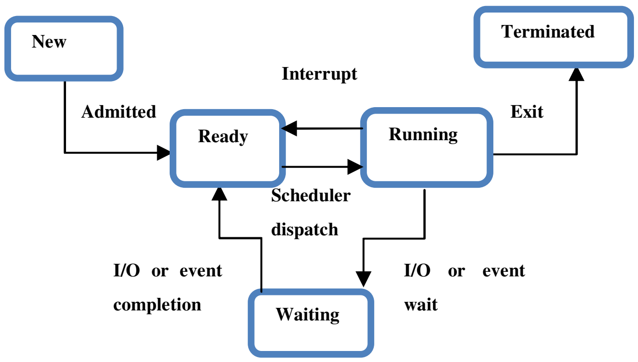


Fig.1. Process States [7]

A. *Algorithm's Performance Evaluation*

CPU scheduling algorithms have a variety of characteristics and the choice of a particular algorithm on the basis of its characteristics may favour one class of processes over another. During the choice of an algorithm, different properties of the various algorithms should be considered as the comparative performance of algorithm depends on a variety of factors as given below:

- UTILIZATION: It is the fraction of time a device is in use. It is measured in the form of ratio of in-use time / total observation time.
- THROUGHPUT: It is the number of job completions in a period of time. The measurement used to calculate this factor is the result of number of jobs / time unit.
- SERVICE TIME: It is the time required by a device to handle a request. It is calculated in the form of specific time unit (exp. seconds).
- QUEUEING TIME: It is the time on a queue waiting for service from the device. It is computed in terms of specific time unit.
- RESIDENCE TIME: It is the time spent by a request at a device. It is calculated by computing the service time and queuing time together.
- RESPONSE TIME: It is the time used by a system to respond to a user Job. It is measured in the form of specific time unit (exp. seconds).
- THINK TIME: Time spent by the user of an interactive system to figure out the next request. It is calculated in terms of specific time unit (exp. seconds).

The objective of these factors is to optimize the algorithm's performance.

II. LITERATURE REVIEW

The fundamental task of an operating system is to allocate the CPU to the jobs for a particular time slice which is termed as job scheduling [1]. During the CPU allocation scheduler and dispatcher is used. Scheduling requires careful attention to ensure fairness and avoid process starvation in the CPU. A variety of scheduling algorithms exist which vary in efficiency according to the jobs to be processed. Scheduling algorithms are generally classified into preemptive and non-preemptive scheduling disciplines [2].

Preemptive Scheduling: The entire running task is interrupted for some time and resumed later when the priority task has finished its execution [10].
Non-Preemptive Scheduling: The entire running task is executed till completion. It cannot be interrupted until terminated.
An overview of most used job scheduling algorithm is discussed below:

a. First Come First Served (FCFS) Scheduling: It is the simplest scheduling algorithm that allocates the CPU to the process that requests the CPU first. This algorithm is easily managed with a FIFO queue [1]. Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline, once a process has a CPU, it runs to completion [3]. FCFS is optimal for smaller processes rather than larger processes [2]. As larger processes occupied/engaged processor for a long time that fallout low throughput.

b. Shortest Job First (SJF) Scheduling: The conclusive factor of SJF scheduling algorithm is, a process with the minimum CPU burst, is served first by the CPU. SJF uses the FCFS to break tie where two processes have the same length next CPU burst. It is provably optimal since it minimizes the average turnaround time and the average waiting time. The main problem with this discipline is the necessity of the previous knowledge about the time required for a process to complete. The SJF algorithm may be implemented in both ways: Preemptive: currently executing process can be preempted when a new process arrives with shortest CPU burst length. This scheme is known as the Shortest -Remaining-Time-First (SRTF) [4]. Non-preemptive: once CPU allocated to a process, no other process can preempt it before its completion [7].

c. Round Robin (RR) Scheduling: This is a preemptive scheduling algorithm, intended mainly for time sharing systems in which a small quantum of time i.e. time slices is assigned to every process. CPU is switched from one process to another process, as the time slice expires [2]. It is designed to give a better responsive but the worst turnaround and waiting time due to the fixed time quantum concept. The scheduler assigns a fixed time unit (quantum) per process usually 10-100 milliseconds, and cycles through them [8]. RR is similar to FCFS except that preemption is added to switch between processes [5].

d. Priority Based Scheduling: In this algorithm, CPU is allocated to the processes on the basis of priority that is associated with each and every process. Usually, lower numbers are used to represent higher priorities. The process with the highest priority is allocated first.If there are multiple processes with same priority, typically the FCFS is used to break tie. This algorithm have starvation problem because sometime low priority processes may never execute. The solution proposed here, named aging: as time progresses increase the priorityof the process, so eventually the process will become the highest priority and will gain the CPU. Priority Based Scheduling [8] can be either implemented in both ways [4]. In preemptive: newly arrived process with higher priority can preempt the currently running process with lower priority. In non-preemptive priority: place the newly arrived highest priority process at the head of the ready queue without any preemption [6].

e. *Multilevel Queue (MQ):* In this algorithm there are several ready queues and scheduling must be done between the multiple queues [2]. Each ready queue is partitioned into separate queues varies foreground (interactive) and background (batch). The foreground ready queue is interactive in nature and used RR algorithm for scheduling purpose but the background is in the form of batch and used FCFS [9]. Here, scheduling used the two ways: Fixed priority scheduling and Time slice.

f. *Multilevel Feedback Queues (MLFQ):* There are several ready queues, each with different priority. When the CPU is available, the scheduler selects a process from the highest-priority, non-empty ready queue. Within a queue [9], it uses RR scheduling. If a process waits too long in a lower-priority queue may be moved to a higher-priority queue (this form of aging to prevent starvation). If a process uses too much CPU time [2], it will be moved to lower-priority queues. This leaves I/O bound and interactive processes in the higher-priority queues.

Aforementioned different scheduling algorithms discuss diverse strategy to process the jobs on the basis of their burst times that can be of preemptive and non-preemptive in nature.

Algorithm Selection Analysis

From the above discussed scheduling algorithms, round robin scheduling algorithm (i.e. preemptive in nature) and first come first serve algorithm ((i.e. non-preemptive in nature)) is selected for analysis further with the proposed algorithm MIN-MAX [10] [11].

III. PROPOSED WORK

As discussed earlier, all algorithms select the processes from ready queue one after another and allocate the CPU to them. But in our new proposed algorithm, the processes are sorted in increasing order of their burst time so that shortest process will remove earlier from the ready queue to give better turnaround time and waiting time. Whenever a process comes, the required burst time is compared with the available processes in the ready queue and accordingly ready queue is updated [7].The following table defines the five processes in a ready queue waiting for CPU allocation:

PROCESSES	P1	P2	P3	P4	P5
BURST TIME	48	16	37	07	23

The following table defines in what manner, our proposed scheduling algorithm arrange these processes in ascending order in a ready queue according to their burst time. This is done [7] with the implementation of insertion sort that updates the ready queue which holds the processes for execution. The following table shows the updated scenario of the ready queue:

PROCESSES	P4	P2	P5	P3	P1
BURST TIME	07	16	23	37	48

MIN-MAX CPU scheduling algorithm is designed to reduce the starvation problem between minimum and maximum burst time processes [12]. Thus, this proposed algorithm results in less overheads viz. starvation, context switching and also improves the average waiting time in contrast to another algorithms.

IV. SIMULATION DESIGN

In order to validate the proposed algorithm over the existing algorithm First Come First Served (FCFS) and Round Robin (RR), we have devised/designed the simulator using the DOT NET platform which offers Graphical User Interface (GUI) to the user. This interface helps the users to input the jobs in ready queue by using the ADD button and the burst time of all the input jobs is decided by randomized function. The Fig.3 mentioned below depicts the home screen window of our simulation design.

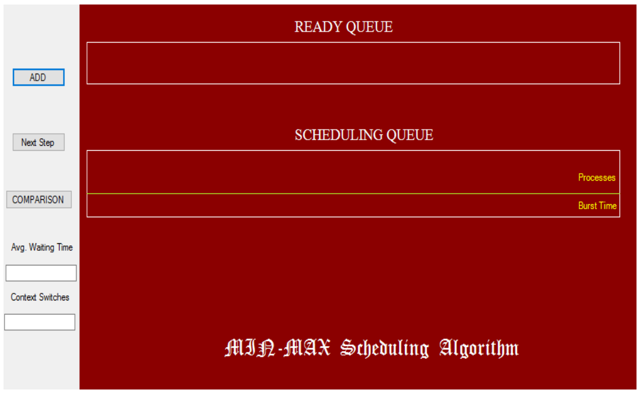


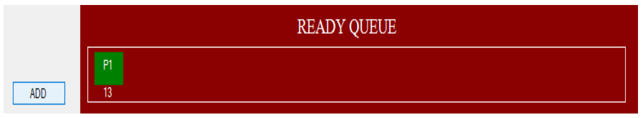
Fig.3. Home Screen Window

This window consists of ready queue (queue of all input processes) and scheduling queue (queue of processes scheduled according to burst time). The ADD button is used to enter the job in ready queue. The ready queue is accompanied with insertion sort an objective to arrange all the jobs in ascending order according to their burst time. After the job insertion phase, the next step is to schedule the jobs for execution. The function of the NEXT button is to schedule and execute the jobs from the ready queue. The job remove earlier from ready queue is scheduled prior in scheduling queue for execution. This process is continuously followed until all the jobs from ready queue are executed completely. On completion of the process's execution COMPARISON function is performed to get comparative analysis of proposed algorithm MIN-MAX, Round Robin algorithm and First Come First Serve algorithm [12]. The average waiting time and context switches parameters are used in comparison states for performance evaluation of different algorithms [13].

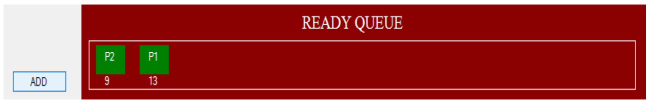
Algorithm Methodology

The proposed scheduling algorithm MIN-MAX follows the minimum-maximum strategy designed to reduce the starvation problem between minimum and maximum burst time processes. The detailed steps of simulation/implementation of proposed algorithm are mentioned below:

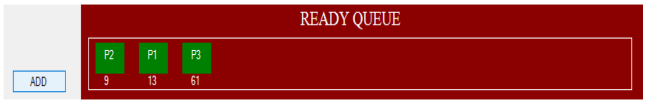
Step-1: Process P1 entered in the ready queue with the burst time 13.



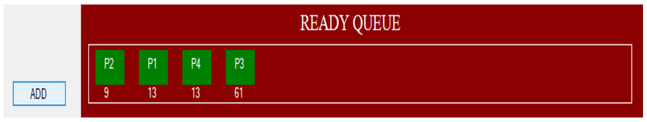
Step-2: Process P2 entered in the ready queue with the burst time 9.



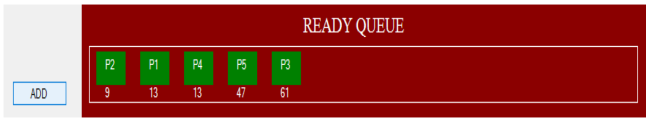
Step-3: Process P3 entered in the ready queue with the burst time 61.



Step-4: Process P4 entered the ready queue with the burst time 13.

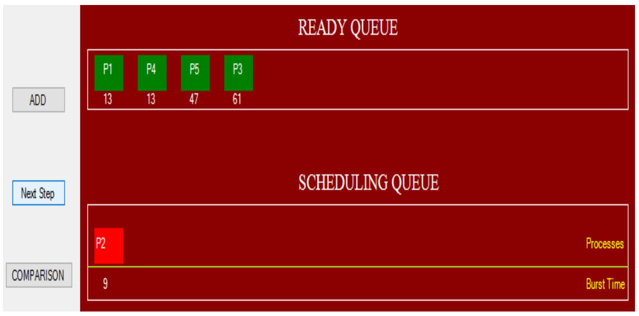


Step-5: Process P5 entered in the ready queue with the burst time 47.

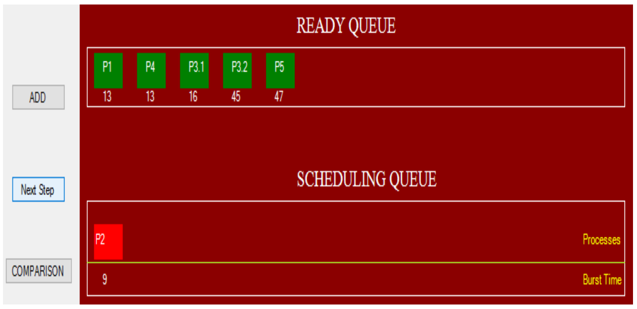


In the aforementioned simulation window, the processes along with their burst times are arranged in ascending order i.e. job with smallest burst time placed at first position and the job with largest burst time placed at last. Sorting of the processes is followed by scheduling and execution of the processes that are placed in ready queue. Scheduling of processes is based on the methodology of proposed algorithm MIN-MAX as discussed below.

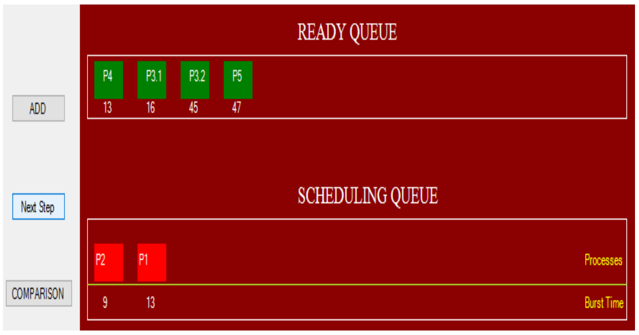
Step-6: Remove process P2 (i.e. from the beginning) from ready queue and schedule it for execution and update the ready queue according to remaining processes as shown in below window.



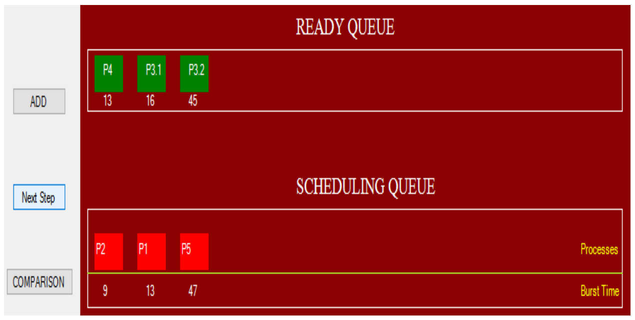
Step-7: Remove process P3 (i.e. from the last) from the ready queue. The burst time of P3 process i.e. 61 is larger than predefined thresh-hold i.e.50. Splits the process P3 into two parts in the ratio of 1/4 and 3/4 and update the ready queue according to remaining processes as shown in below window.



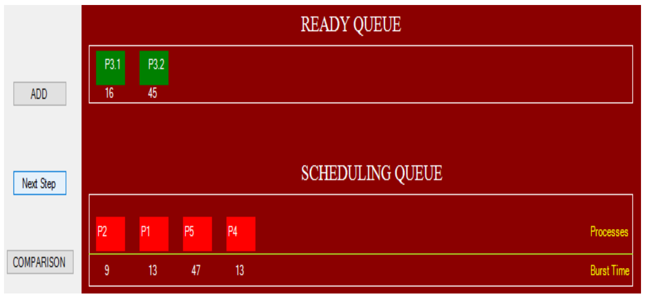
Step-8: Remove process P1 (i.e. from the beginning) from the ready queue and schedule it for execution and update the ready queue for the rest of processes.



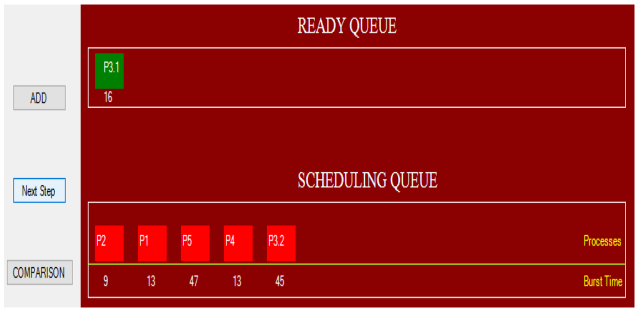
Step-9: Remove process P5 (i.e. from the last) from the ready queue and schedule it for execution and update the ready queue for the rest of processes.



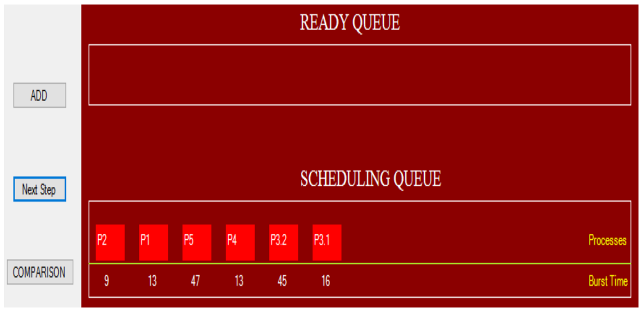
Step-10: Remove process P4 (i.e. from the beginning) from the ready queue and schedule it for execution and update the ready queue for the remaining processes.



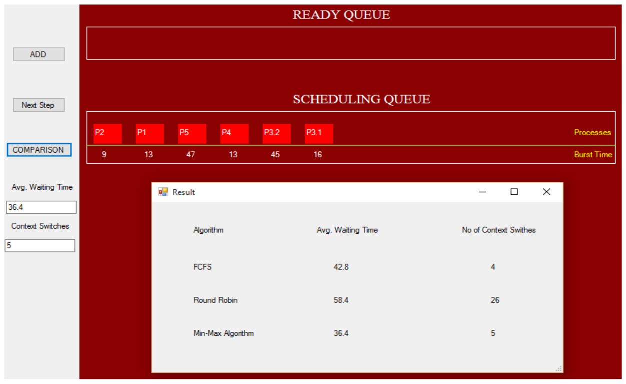
Step-11: Remove process P3.2 (i.e. from the last) from the ready queue and schedule it for execution.



Step-12: Remove the one and only one process P3.1 from the ready queue and schedule it for execution.



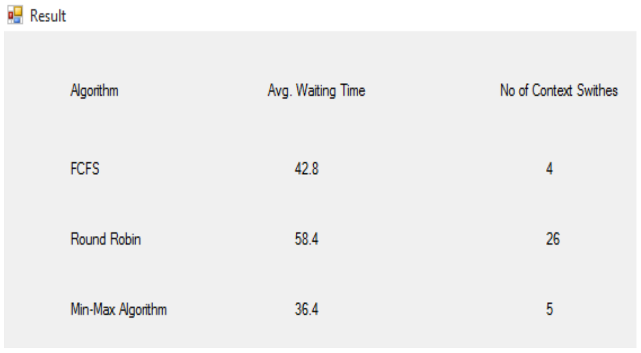
The successful execution of all processes is completed by using ready queue (i.e. handles all processes with their burst times in ascending order) and scheduling queue (i.e. executes all the processes from the ready queue based on the threshold value). Further performance of the scheduling algorithm is evaluated on the basis of two parameters: average waiting time and context switches. For this purpose, COMPARISON function is applied as depicts in the following window that explains about the performance of the proposed algorithm MIN-MAX Round Robin algorithm and First Come First Serve algorithm.



Algorithm	Avg. Waiting Time	No of Context Swithes
FCFS	42.8	4
Round Robin	58.4	26
Min-Max Algorithm	36.4	5

V. EXPERIMENTAL RESULTS

The proposed scheduling algorithm MIN-MAX is both preemptive and non-preemptive in nature [13]. This algorithm is devised on a *thresh-hold* vector that decides either to preempt the running process or to not. All the processes along with their burst times sorted in ascending order and the shortest process is removed and processed earlier from the ready queue to get better turnaround time and waiting time. The entire scheduling algorithm is divided into iterations and each iteration performed two steps except the 4th, 8th, 12th, 16th …… so on iterations perform only single step (i.e. specifically execute the middle job from remaining jobs) to trim down the starvation problem of middle process. The key objective of proposed scheduling algorithm is to reduce the starvation problem between minimum and maximum burst time processes.



Algorithm	Avg. Waiting Time	No of Context Swithes
FCFS	42.8	4
Round Robin	58.4	26
Min-Max Algorithm	36.4	5

Therefore, this proposed algorithm results in less overheads viz. starvation, context switching and also improves the average waiting time in contrast to another algorithms.

VI. MIN-MAX ALGORITHM COMPLEXITY

The complexity of our proposed algorithm in terms of Big – Oh notation as given below:

	Complexity: Big-Oh	
Best case	Average case	Worst case
O (c)	O (n)	O (n)

VII. COMPARITIVE STUDY

The below mentioned table depicts the comparative study [7] of a variety of scheduling algorithms viz. First Come First Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Round Robin (RR) and MIN-MAX on the basis of selection function, decision mode, throughput, response time, overhead, effect on processes and starvation [2].The selection function finds out which process is to be selected next for execution among ready processes. This function can be characterized by priority, resource requirements, or the execution characteristics of the process. The selection function parameter is signified by following three quantities:
s: service time,
w: waiting time,
e: execution time

Algorithms / Parameters	FCFS	SJF	SRT	Round Robin	MIN-MAX
Selection function	Max[w]	Min[s]	Min[s-e]	constant	Min[s],max[s] In alternate manner
Decision mode	Non-preemptive	Non-preemptive	Preemptive	Preemptive (at time quantum)	Preemptive and Non-preemptive
Throughput	Not emphasized	High	High	Depends on time quantum	Average
Response time	Depends on process execution time	Suitable for shortest burst time processes	Provide good response time	Suitable for shortest burst time processes	Average
Overhead	Minimum	Can be high	Can be high	Minimum	Minimum
Effect on processes	Penalized short processes: I/O bound processes	Penalized long Processes	Penalized long Processes	Fair treatment	Good balance
Starvation	NO	Possible	Possible	NO	NO

Table 2: a comparative study [7] of scheduling algorithms

VIII. CONCLUSION

This manuscript is based on the analysis and comparison of three proportional-shares CPU scheduling algorithm for single core machines. The simulation of proposed CPU scheduling algorithm helps in improving the performance of first come first serve and round robin algorithm as discussed in the experimental results section. The objective of this research work is to reduce the starvation problem between minimum and maximum burst time processes and to attain this objective- threshold vector is used as mentioned in algorithm methodology. To demonstrate the performance sensitivity, two relatively simple system parameters viz. average waiting time and context switches are used. We have discussed the scenario that could be a simple step for a huge aim in obtaining an optimal scheduling algorithm. The proposed simulation design is also useful for selection of better job scheduling algorithm with respect to different scenario as a view point of system performance.

REFERENCES

[1]. Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts", John Wiley & Sons, Inc., Seventh (7th)Edition, ISBN: **0-471-69466-5**, **2005.**

[2]. William Stallings, "Operating Systems: Internals and Design Principles", Sixth (6th) Edition, **2009**.

[3]. L. Yang, J. M. Schopf and I. Foster, "Conservative Scheduling: Using predictive variance to improve scheduling decisions in dynamic environments", In Proceedings of the ACM/IEEE conference on Supercomputing, Page No (**15- 31**), **2003**.

[4]. Milam Milenkovic, "Operating Systems Concepts and Design", McGraw-Hill, Computer Science Series, Second (2nd) Edition.

[5]. 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, Volume-**06**, Issue-**10**, **2009**.

[6]. Baer, J. L., "A survey of some theoretical aspects of multiprocessing", Computing Surveys, Page No (**31-80**), **1973**.

[7]. K. Sukhija, N. Aggarwal and M. Jindal, "An Optimized approach to CPU Scheduling Algorithm: Min-Max", Journal of Emerging Technologies in Web Intelligence, Volume-**06**, Issue-**04**, Page No **(420-428)**, **2014**.

[8]. S. Raheja, R. Dhadich and S. Rajpal, "An Optimum time Quantum using Linguistic Synthesis for Round Robin CPU Scheduling Algorithm" , International Journal on Soft Computing , Volume-**03**, Issue-**01**, **2012**.

[9]. Andrew S. Tanenbaum, and Albert S. Woodfhull, "Operating Systems Design and Implementation", Second (2nd) Edition, **2005**.

[10]. H.H.S. Lee; Lecture: CPU Scheduling, School of Electrical and Computer Engineering, Georgia Institute of Technology.

[11]. E. O. Oyetunji and A. E. Oluleye, “Performance Assessment of Some CPU Scheduling Algorithms”, Research Journal of Information Technology, Volume-**01**, Issue-**01**, Page No (**22-26), 2009**.

[12]. S. Babu, N Priyanka and P.Suresh, “A Novel CPU Scheduling Algorithm–Preemptive &Non-Preemptive”, International Journal of Modern Engineering Research, Volume-**02**, Issue-**06**, Page No (**4484-4490), 2012**.

[13]. L Cherkasova, Diwaker G and A. Vahdat, “Comparison of the Three CPU Schedulers in Xen”, SIGMETRICS Performance Evaluation Review, Volume-**35**, Issue-**02**, Page No **(42-51), 2007**.

AUTHOR PROFILE

Researcher Karan Sukhija has done his bachelor degree in computer science and application at RSD College, Ferozepur City (Batch 2006-09). He was Completed his Masters in Computer science and application at department of computer science and application, panjab university, chandigarh (Batch 2009-12). Currently he is doing research in the area of Educational Data Mining at Panjab university, Chandigarh.