IMPROVED LONGEST JOB FIRST CPU SCHEDULING ALGORITHM (ILJF)

By

AISHA ALIYU SHEHU

DEPARTMENT OF MATHEMATICS

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

AUGUST, 2016.

IMPROVED LONGEST JOB FIRST CPU SCHEDULING ALGORITHM

By

Aisha Aliyu SHEHU

B.Tech Maths/Computer Science (FUT Minna), 2010

MSc/SCI/11848/2011-2012

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF MATHEMATICS,

FACULTY OF SCIENCE

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

AUGUST, 2016

DECLARATION

I declare that the work in this Dissertation entitled "*Improved Longest Job First (ILJF) CPU Scheduling Algorithm*" has been performed by me in the Department of Mathematics under the supervision of Professor S.B Junaidu and Dr. S.E Abdullahi.

The information derived from the literature has been duly acknowledged in the text and list of references provided. No part of this Dissertation was previously presented for another degree or diploma at any university.

Aisha Aliyu SHEHU

_____	_____	_____
Name of student	Signature	Date

CERTIFICATION

This Dissertation entitled IMPROVED LONGEST JOB FIRST (ILJF) CPU SCHEDULING ALGORITHM by AISHA ALIYU SHEHU meets the regulation governing the award of the degree of M.Sc. Computer Science of Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

_____ (Signature): _____

Chairman, Supervisory Committee Date: _____

_____ (Signature): _____

Member, Supervisory Committee Date: _____

_____ (Signature): _____

Member, Supervisory Committee Date: _____

_____ (Signature): _____

Head of Department Date: _____

_____ (Signature): _____

Dean, School of Postgraduate Studies Date: _____

DEDICATION

I dedicate this work to Almighty Allah (S.W.A), for seeing me through in all things. And also to my beloved parents in the name of DR A.Y. Shehu and Hajiya Fatima Shehu for not only their financial support but for the time and resources they spent in training and bringing me up to be a person of value and importance to the society and life in general.

ACKNOWLEDGEMENT

In the name of Allah, the Most Gracious, the Most Merciful. All praise and thanks are Allah's for giving me the health and intellect to see the actualization of the work. My humble salutation to my Noble prophet Muhammad (S.A.W) and members of his household.

First and foremost, I want to register my appreciation with thanks and due respect to my humble supervisors Professor S.B Junaidu (Major) and Dr. S.E Abdullahi (Minor) for supervising this work. In fact, you did more than supervisors will ordinarily do. For their positive comments, suggestions and general concern which finally brought this work to this stage. I am immensely thankful. I also acknowledge the effort of the entire lecturers of the Department of Mathematics, Ahmadu Bello University Zaria, for their academic and intellectual support towards the success of this Dissertation.

A special acknowledgment goes to my family for their moral and financial support for easy running of my entire study programme. I cannot forget mentioning some of my colleagues who contributed one way or the other towards the completion of this research work, people like Abdulrazaq. A, Hajara I, Alhassan, Teema, Fatima Ridwan, Ibrahim, MZ, Bola and my entire class members.

My sincere gratitude to Dr Bashir, and Dr. F.A Kana of Department of Mathematics, Ahmadu Bello University, Zaria for their guidance, ideas and support throughout this research work.

A special acknowledgment also goes to my husband; Aliyu Muhammad K. for his academic, moral and financial support towards the successful completion of this Dissertation.

Finally, I thank you all. May Almighty Allah (S.W.A) be with you all in all your endeavors. Ameen

ABSTRACT

Longest Job First (LJF) is a Scheduling algorithm that assigns processes with longer burst times first before processes with shorter burst times. This often leads to starvation of the shorter processes which in turn affects the performance of the system. An earlier research extended the LJF algorithm with a *combinational burst time* to curtail the starvation of short processes. However, performance metrics (Average Waiting Time, Average Turn Around Time and Average Response time) were not fully optimized using this combinational burst time extension. After careful study, a new algorithm was proposed (called Improved LJF Scheduling Algorithm) which overcomes limitations of LJF algorithm with combinational burst time. The proposed algorithm was implemented and bench marked against five scheduling algorithms. Results of our experiments showed that the proposed algorithm outperformed other scheduling algorithms by up to 65% for systems that adopt LJF CPU Scheduling Algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ART: Average Response Time

ATAT: Average Turn-Around Time

AWT: Average Waiting Time

BW: Burst Time Priority Weight

CBT: Combinational Burst Time

CPU: Central Processing Unit

CS: Context Switch

CWA: Combined Weighted Average

FCFS: First Come First Served

FIFO: First in First Out

I/O: Input Output

LJF: Longest Job First

LJF+CBT: Longest Job First + Combinational Burst Time

PCB: Process Control Block

SPT: Shortest Processing Time

OS: Operating System

TQ: Time Quantum

TAT: Turn-Around Time

WT: Waiting Time

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

An **operating system** (**OS**) is a software system that manages computer hardware and software resources. It provides common services for computer programs and acts as an intermediary between the computer user and the computer hardware (Silberschatz *et al.,* 2005). It is an essential component of the system software in a computer system. It can be found on almost any device that contains a computer—from cellular phones and video game consoles to supercomputers and web servers. One of the most important management unit of OS amidst memory management unit, storage management unit etc. is the process management unit. The process management unit allocates processes to be executed to the processor. This process management unit is used to achieve an important aspect of the OS, which is multiprogramming. Multiprogramming increases central processing unit utilization by organizing processes (code and data), so that the CPU always have one to execute at any time (Stallings, 2012).

Multiprogramming requires several processes to be kept simultaneously in memory. Since in general, the main memory is too small to accommodate all processes, the processes are kept initially on the disk in the process pool. This pool consists of all processes residing on the disk, awaiting allocation of main memory. When the operating system selects a process from the process pool, it loads that process into memory for execution. If several processes are ready to run at the same time, the operating system must choose among them; making this decision is CPU scheduling (Silberschatz *et al.,* 2005). CPU scheduling is a necessary operating system task; therefore, its scheduling is central to operating system design. When there is more than one process in the ready

queue awaiting its turn to be assigned to the CPU, the operating system must decide through the scheduler the order of execution.

1.2 Research Motivation

Longest Job First (LJF) CPU scheduling algorithm which assigns the CPU to the process in queue with the highest next CPU burst suffers from the problem of starvation of processes with short burst times. (Abdullahi and Junaidu, 2013) made an improvement to the LJF CPU scheduling algorithm by minimizing the starvation problem. With this improvement, however, performance metrics were not fully optimized. The motivation for this research work was to propose an algorithm that overcomes poor Average Waiting Time (AWT), Average Turn Around Time (ATAT), Average Response Time (ART) in the Longest Job First with Combinational Burst time scheduling algorithm by (Abdullahi and Junaidu, 2013)

1.3 Research Aims and Objectives

The aim of this research is to enhances the longest job first CPU scheduling algorithm. This was done by proposing a new algorithm that modified the Longest Job First with Combinational Burst Time scheduling algorithm of (Abdullahi and Junaidu, 2013)

The following are the objectives of this research work:

a. Propose an algorithm based on the modification of Longest Job First+Combinational Burst Time Scheduling algorithm by (Abdullahi and Junaidu, 2013)
b. Evaluate the algorithm analytically and through simulation side by side with First Come First Serve, Shortest Job First, Longest Job First, Round Robin, Longest Job First with Combinational Burst Time.

1.4 Research Methodology

The following steps were taken to actualize this work:

a. Intensive study of CPU scheduling algorithms
b. Review of literature on Longest Job First (LJF) scheduling algorithm
c. Modify the LJF+CBT scheduling algorithm by using percentile in deciding the threshold for the categorization of short processes to be merged
d. Simulate the modified algorithm using Java programming
e. Evaluate the modified algorithm at different percentile threshold and compare the results with FCFS, SJF, LJF, RR, LJF+CBT based on Average Waiting Time, Average Turn Around Time, Average Response Time.
f. Generation of data used in the simulation: the burst times of the processes were generated using three different distributions as follows:
    i. Uniform distribution: It defines equal probability over a given range for a continuous distribution. For this reason, it is important as a reference distribution and also used in the generation of random numbers. That is, almost all random number generators generate random numbers on the (0,1) interval. For other distributions, some transformation is applied to the uniform random numbers.
    ii. Normal distribution: It is the most widely known and used of all distributions. It approximates many natural phenomena so well; it has developed into a standard of reference for many probability problems.
    iii. Exponential distribution: It is used to model the behavior of units that have a constant failure rate.

1.5 Contribution to Knowledge

The research contributed to the existing knowledge as follows:

a. Improvement of the LJF scheduling algorithm which when applied to the LJF+CBT scheduling algorithm, increases the performance of the LJF scheduling algorithm by minimizing the average waiting time, average turnaround time and average response time positively
b. It also minimizes starvation of processes in Longest Job First (LJF) scheduling algorithm through merging shorter jobs (Combinational Burst Time, CBT) to maximize total utilization of the CPU.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview of Operating System

An operating system interacts between the user and the computer hardware. The purpose of an operating system is to provide a platform in which a user can execute programs in well-located and efficient manner (Silberschatz *et al.,* 2005). Modern operating systems and time sharing systems are more complex, they have evolved from a single task to a multitasking environment in which processes run in a synchronized manner. CPU scheduling is a necessary operating system task; therefore, its scheduling is central to operating system design. When there is more than one process in the ready queue or job pool waiting its turn to be assigned to the CPU, the operating system must decide through the scheduler the order of execution. Allocating CPU to a process requires careful awareness to assure justice and avoid process starvation for CPU. Scheduling decisions try to reduce the following: turnaround time, response time and average waiting time for processes and the number of context switches (Saroj and Roy, 2012).

2.2 Process

A process is an instance of an executing program, including the current values of the program counter, registers and variables (Tanenbaum, 2009). The execution of a process must progress in a sequential fashion, which is referred to as process state as shown in figure 2.1. Components of a process are:

a. **New:** The process is being created.

b. **Ready:** The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run.

c. **Waiting:** The process is waiting for some event to occur (such as the completion of an I/O operation).

d. **Terminated:** The process has finished execution.

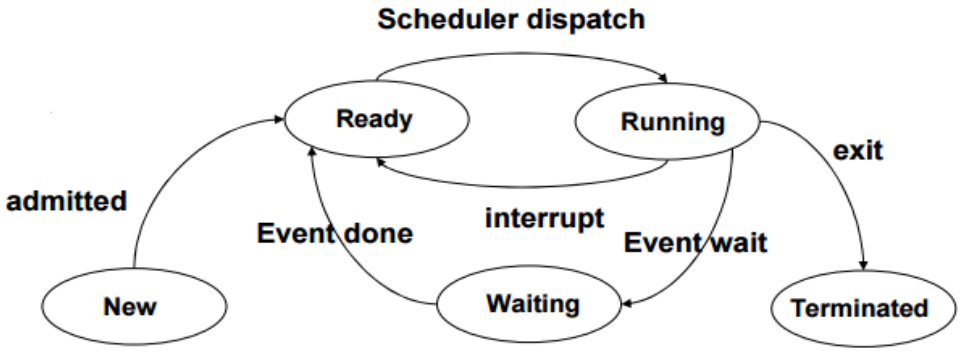


Figure 2. 1: Diagram of Process State (Frédéric, 2007)

2.3 Process Control Block (PBC)

Each process is represented in the operating system by a process control block (PCB) also called a task control block (Silberschatz *et al.,* 2005). A PCB contains many pieces of information associated with a specific process (figure 2.2) which is described below

a. **Pointer:** Pointer points to another process control block. Pointer is used for maintaining the scheduling list.
b. **Process state:** Process state may be new, ready, running, waiting and so on.
c. **Program counter:** Program Counter indicates the address of the next instruction to be executed for this process.
d. **CPU registers:** CPU registers include general purpose register, stack pointers, index registers and accumulators. Number of register and type of register totally depends upon the computer architecture.

e. **Memory management information:** This information may include the value of base and limit registers, the page tables, or the segment tables depending on the memory system used by the operating system. This information is useful for deallocating the memory when the process terminates.

f. Accounting information: This information includes the amount of CPU and real time used, time limits, job or process numbers and account numbers.

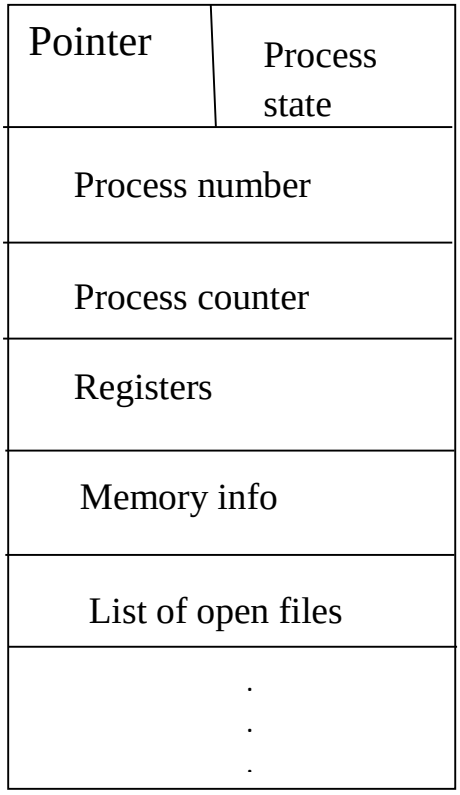


Figure 2. 2: Information in Process Control Block **(**Silberschatz *et al*., 2005**)**

Process control block includes CPU scheduling, I/O resource management and file management information. The PCB serves as the repository for any information which can vary from process to process. Loader/linker sets flags and registers when a process is created. If that process gets suspended, the contents of the registers are saved on a stack and the pointer to the particular stack frame is stored in the PCB. By this technique, the hardware state can be restored so that the process can be scheduled to run again.

2.3.1 Process scheduling

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy (Silberschatz *et al.*, 2005).

Process scheduling is an essential part of a Multiprogramming operating system. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded processes share the CPU using time multiplexing.

2.4 Scheduling Queues

Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. (Silberschatz *et al*., 2005)

Figure 2.3 shows the queuing diagram of process scheduling.

a. Queue is represented by rectangular box.
b. The circles represent the resources that serve the queues.
c. The arrows indicate the process flow in the system.

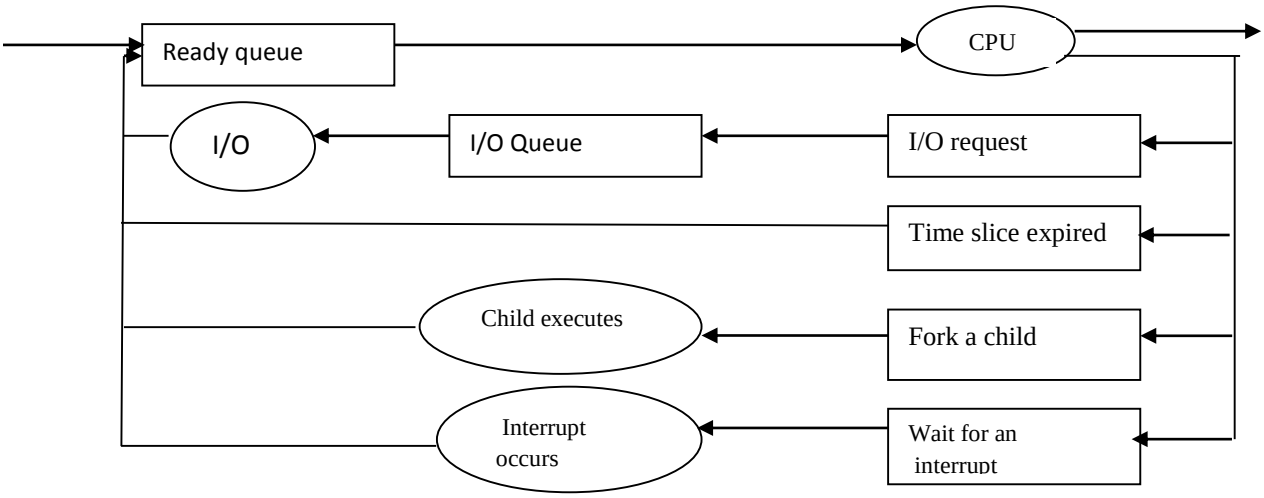


Figure 2. 3: Queuing Diagram of Process Scheduling (Silberschatz *et al.*, 2005)

A newly arrived process is put in the ready queue. Processes waits in ready queue for allocation of the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur.

a. The process could issue an I/O request and then it would be placed in an I/O queue.
b. The process could create new sub process and will wait for its termination.
c. The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue.

In the first two cases, the process eventually switches from the waiting state to the ready state and is then put back in the ready queue. A process continues this cycle until it terminates, at which it is removed from all queues and has its PCB resources deallocated (Silberschatz *et al.,* 2005)

2.5 Schedulers

Schedulers are special system software which handles process scheduling in various ways (Silberschatz *et al*., 2005). Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types

a. Long Term Scheduler
b. Short Term Scheduler
c. Medium Term Scheduler

2.5.1 Long Term Scheduler

It is also called job scheduler. Long term scheduler determines which programs are admitted to the system for processing. Job scheduler selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling. The primary objective of the

job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound. It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system. On some systems, the long term scheduler may not be available or minimal. Time-sharing operating systems have no long term scheduler. When process changes the state from new to ready, then there is use of long term scheduler.

2.5.2 Short Term Scheduler

It is also called CPU scheduler. Main objective is increasing system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects process among the processes that are ready to execute and allocates CPU to one of them.

Short term scheduler also known as dispatcher, execute most frequently and makes the fine grained decision of which process to execute next. Short term scheduler is faster than long term scheduler.

2.5.3 Medium Term Scheduler

Medium term scheduling is part of the swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium term scheduler is in-charge of handling the swapped out-processes.

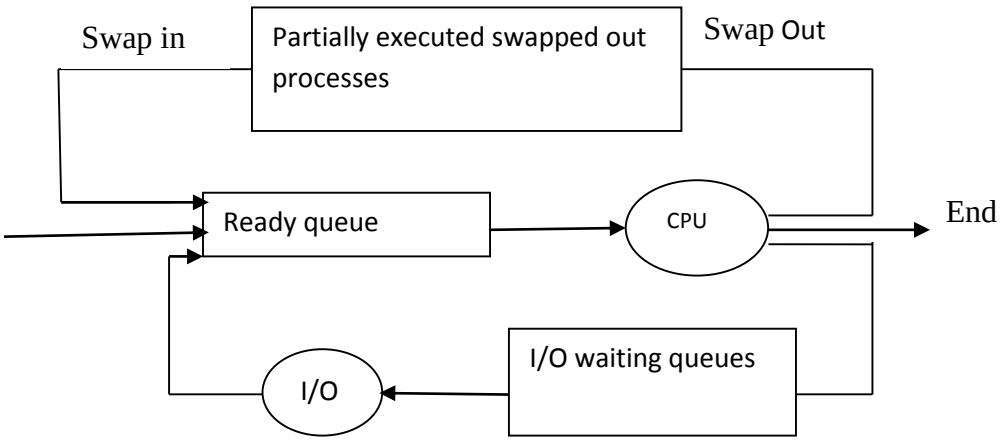


Figure 2. 4: Medium-Term Scheduler (Silberschatz *et al.*, 2005)

Running process may become suspended if it makes an I/O request. Suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other process, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

2.6 Scheduling Algorithms

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated to the CPU next. There are many Scheduling Algorithms. Some basic scheduling algorithms are:

a. First Come First Serve (FCFS) Scheduling
b. Shortest-Job-First (SJF) Scheduling
c. Priority Scheduling
d. Round Robin (RR) Scheduling
e. Longest job first (LJF) Scheduling

2.6.1 First-Come First-Serve (FCFS)

It is the simplest scheduling policy also known as first in first out(FIFO) or strict queuing scheme. As each processes become ready, it joins the queue. When the currently running process ceases to execute the process that has been in the ready queue the longest is selected for running. (Stalling, 2012). To illustrate the FCFS scheduling, Consider the following set of processes that arrive at time 0, with their burst times given in milliseconds as shown in Table 2.1

Table 2. 1: Process Table (Silberschatz *et al.*, 2005)

Process	Burst Time
P_1	24
P_2	3
P_3	3

If the processes arrive in the order *P*1, *P*2, *P*3, and are served in FCFS order, the result is shown in the Gantt chart in Figure 2.5

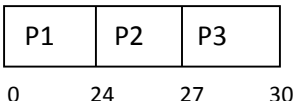


Figure 2. 5: Gantt chart for FCFS (Silberschatz *et al.*, 2005)

The waiting time is 0 milliseconds for process *P*1, 24 milliseconds for process *P*2, and 27 milliseconds for process *P*3. Thus, the average waiting time, average turnaround time and average response time is shown in Table 2.2

Table 2. 2: FCFS Result (Silberschatz *et al.*, 2005)

Waiting time	Turnaround Time	Response Time
P1=0 P2=24 P3=27	P1=24 P2=27 P3=30	P1=0 P2=24 P3=27
AWT= 17ms	ATAT= 27ms	ART=17ms

Advantages

a. It performs much better for longer processes than shorter ones (Stallings, 2012)

b. Easy to understand and implement. (Silberschatz *et al*., 2005)

Disadvantages

a. Convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU. Short process behind long process results in lower CPU utilization (Silberschatz *et al*., 2005; Rahul and Umesh, 2012).

b. It tends to favor processor-bound processes over I/O bound processes (Stallings, 2012)

2.6.2 Shortest-Job-First (SJF)

This algorithm associates with each process the length of the next process's CPU burst time. When the CPU is available, it is assigned the process that has the least next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie. The SJF can be either preemptive or non-preemptive. The choice arises when a new process arrives in the ready queue while a previous process is still executing. The next CPU burst of the newly arrived process may be shorter than the time remaining in the process whose burst is currently on the CPU. A preemptive SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst (Oyetunji and Oluleye, 2009; Silberschatz *et al*., 2005). Preemptive SJF is sometimes referred to as shortest remaining time first scheduling (Silberschatz *et al*., 2005; Stallings, 2012).

To illustrate the SJF scheduling, consider the following set of processes, with the length of their burst times given in milliseconds as shown in Table 2.3

Table 2. 3: Process Table (Silberschatz *et al.*, 2005)

Process	Burst Time
*P*1	6
*P*2	8
*P*3	7
*P*4	3

Using SJF scheduling, these processes are scheduled according to the Gantt chart in Figure 2.6

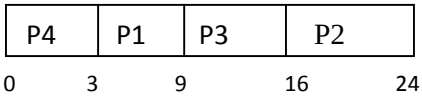


Figure 2. 6: Gantt chart for SJF (Silberschatz *et al.*, 2005)

The waiting time is 3 milliseconds for process *P*1, 16 milliseconds for process *P*2, 9 milliseconds for process *P*3, and 0 milliseconds for process *P*4. Thus, the average waiting time, average turnaround time and average response time is shown in Table 2.4

Table 2. 4: SJF Result (Silberschatz *et al.*, 2005)

Waiting time	Turnaround Time	Response Time
P1=3 P2=16 P3=9 P4=0	P1=9 P2=24 P3=16 P4=3	P1=9 P2=16 P3=9 P4=0
AWT= 7ms	ATAT= 13ms	ART=7ms

Advantages

a. Best approach to minimize average waiting time (Stallings, 2012)

b. Throughput is high (Silberschatz *et al.*, 2005)

Disadvantages

a. Long running processes may starve, because the CPU has a steady supply of short processes (Suri and Sumit, 2012; Silberschatz *et al*., 2005; Rahul and Umesh, 2012; Insup, 2007).

b. It cannot be implemented because of the difficulty in determining the shortest remaining burst time next (Insup, 2007)

2.6.3 Priority Scheduling(PS)

The SJF algorithm is a special case of the general priority scheduling in which the priority is based on shorter jobs. The larger the CPU burst time, the lower the priority of that process and vice versa. The PS algorithm associates with each process a priority and the CPU is allocated to process with the highest priority. Usually, lower numbers are used to represent highest priority but this is not a rule as higher numbers can as well be used. For effective utilization of CPU, Priorities to various processes can be set by a study of type and time of processes during a day. For example, there can be more number of LOGON processes during the day time in an organization and more backup jobs during evening. So, logon processes should be given top priority in the morning hours, where as backup jobs should be given higher priority than other jobs during closing hours. 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. (Silberschatz *et al*., 2005)

To illustrate the Priority scheduling, consider the following set of processes, assumed to have arrived at time 0 in the order *P*1, *P*2, *P3*, *P4*, *P*5, with the length of the CPU burst given in milliseconds as shown in Table 2.5

Table 2. 5: Process Table (Silberschatz *et al.*, 2005)

Process	Burst Time	Priority
*P*1	10	3
*P*2	1	1
*P*3	2	4
*P*4	1	5
*P*5	5	2

Using priority scheduling, these processes are scheduled according to the Gantt chart in Figure 2.7

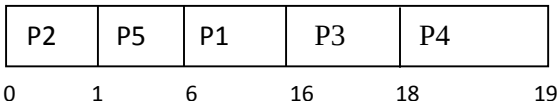


Figure 2. 7: Gantt chart for Priority Scheduling (Silberschatz *et al.*, 2005)

Thus, the average waiting time, average turnaround time and average response time is shown in Table 2.6

Table 2. 6: Priority Scheduling Result (Silberschatz *et al.*, 2005)

Waiting time	Turnaround Time	Response Time
P1=6 P2=0 P3=16 P4=18 P5=1	P1=16 P2=1 P3=18 P4=19 P5=5	P1=6 P2=0 P3=16 P4=18 P5=1
AWT= 8.2ms	ATAT= 14.7ms	ART=8.2ms

Advantage

a. Good response for the highest priority processes (Abdulrazaq *et al*., 2014).

Disadvantage

a. A major problem with priority scheduling is known as indefinite blocking, or starvation, in which a low-priority task can wait forever because there are always some other processes around that have higher priority (Silberschatz *et al*., 2005; Abdulrazaq *et al*., 2014)

2.6.4 Round Robin Scheduling (RR)

Round Robin scheduling is designed for time-sharing systems. It is similar to FCFS scheduling, but preemption is added to the switch between processes. A small time unit called the time quantum or time slice is defined. The ready queue is maintained as a circular queue. The CPU scheduler goes round the ready queue, allocating the CPU to each process for a time interval of up to 1-time quantum. To implement the Round Robin scheduling, we keep the ready queue as a First-In-First-Out (FIFO) queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1-time quantum, and dispatches the process (Silberschatz *et al.*, 2005). One of two things will then happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1 time quantum, the timer goes off, and will cause interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process at the head of the ready queue. In the Round Robin scheduling algorithm, no process is allocated the CPU more than 1 time quantum in a row (unless it is the only runnable process). If a process's CPU burst exceeds 1 time quantum, that process will be

preempted and is put in the ready queue. The Round Robin scheduling algorithm is thus preemptive. (Silberschatz *et al.*, 2005).

To illustrate the RR scheduling. Consider the following set of processes that arrive at time 0, with their burst times given in milliseconds as shown in Table 2.7

Table 2. 7: Process Table (Silberschatz *et al.*, 2005)

Process	Burst Time
*P*1	24
*P*2	3
*P*3	3

Using Round Robin scheduling, these processes are scheduled according to the Gantt chart in Figure 2.8:

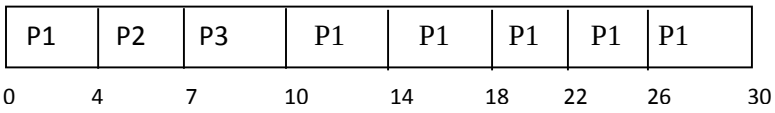


Figure 2. 8: Gantt chart for RR (Silberschatz *et al.*, 2005)

If we use a time quantum of 4 milliseconds, then process *P*1 gets the first 4milliseconds. Since it requires another 20 milliseconds, it is preempted after the first time quantum, and the CPU is given to the next process in the queue, process *P*2. Process *P*2 does not need 4 milliseconds, so it quits before its time quantum expires. The CPU is then given to the next process, process *P*3. Once each process has received 1 time quantum, the CPU is returned to process *P*1 for an additional time quantum. To calculate the average waiting time for the above schedule. *P*1 waits for 6 milliseconds (10 - 4), *P*2 waits for 4 milliseconds, and *P*3 waits for 7 milliseconds. Thus, the average waiting time, average turnaround time and average response time is shown in Table 2.8

Table 2. 8: RR Result (Silberschatz *et al.*, 2005)

Waiting time	Turnaround Time	Response Time
P1= (10-4) =6 P2=4 P3=7	P1=30 P2=7 P3=10	P1=0 P2=4 P3=7
AWT= 5.66 ms	ATAT= 15ms	ART= 3.3ms

Advantages

a. Round-Robin is particularly effective in a general-purpose, time-sharing system or transaction-processing system. (Stallings, 2012).
b. Fair treatment for all the processes

Disadvantages

a. Processor-bound processes tend to receive an unfair portion of processor time which result in poor performance for I/O bound processes, inefficient use of I/O devices and an increase in variance of response time. (Stallings, 2012).
b. If time quantum is too large, RR Scheduling degenerates to FCFS policy (Silberschatz *et al.*, 2005)

2.6.5 Longest Job First(LJF)

In this strategy the scheduler arranges processes with the Burst times in the ready queue, so that the process with high burst time is scheduled first. If two processes have same burst time and arrival time, then FCFS procedure is followed. Its major limitation is that the shorter jobs are *starved* (waiting indefinitely) as longer jobs hog unto the CPU endlessly (Abdullahi and Junaidu, 2013).

To illustrate the LJF scheduling, consider the following set of processes, with their burst times given in milliseconds as shown in Table 2.9

Table 2. 9: Process Table (Silberschatz *et al.*, 2005)

Process	Burst Time
*P*1	6
*P*2	8
*P*3	7
*P*4	3

Using LJF scheduling, these processes are scheduled according to the Gantt chart in Figure 2.9

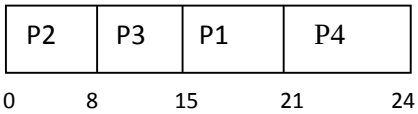


Figure 2. 9: Gantt chart for LJF (Silberschatz *et al.*, 2005)

The waiting time is 15 ms for process *P*1, 0 ms for process *P*2, 15 ms for process *P*3, and 21 ms for process *P*4. Thus, the average waiting time, average turnaround time and average response time is shown in Table 2.10

Table 2. 10: LJF Result (Silberschatz *et al.*, 2005)

Waiting time	Turnaround Time	Response Time
P1=15 P2=0 P3=8 P4=21	P1=21 P2=8 P3=15 P4=24	P1=15 P2=0 P3=8 P4=21
AWT= 11ms	ATAT= 17ms	ART=11ms

Advantage

a. It tries to reduce overall turnaround time and maximize system utilization for high system loads. (Abdullahi and Junaidu, 2013)

Disadvantage

a. Starvation of shorter jobs (Cheng and Kahlbacher, 2006)

There are various CPU scheduling algorithms which have different properties, and the choice of a particular algorithm may favor one class of processes over another. For selection of an algorithm for a particular situation, we must consider properties of various algorithms. The scheduling criteria include the following (Silberschatz *et al.*, 2005)

a. Context Switch: A context switch is process of storing and restoring context (state) of a preempted process, so that execution can be resumed from same point at a later time. Context switching is usually computationally intensive, lead to wastage of time and memory, which in turn increases the overhead of scheduler, so the design of operating system is to optimize only these switches.
b. Throughput: Throughput is defined as number of processes completed per unit time. Throughput is slow in round robin scheduling implementation. Context switching and throughput are inversely proportional to each other.
c. CPU Utilization: This is a measure of how much busy the CPU is. Usually, the goal is to maximize the CPU utilization.
d. Turnaround Time: Turnaround time refers to the total time which is spend to complete the process and is how long it takes the time to execute that process. The time interval from the time of submission of a process to the time of completion is the turnaround time. Total

turnaround time is the sum of the periods spent waiting to get into memory, waiting time in the ready queue, execution time on the CPU and doing I/O.

e. Waiting Time: Waiting time is the total time a process has been waiting in ready queue. The CPU scheduling algorithm does not affect the amount of time during which a process executes or does input-output; it affects only the amount of time that a process spends waiting in ready queue.

f. Response Time: In an interactive system, turnaround time may not be best measure. Often, a process can produce some output fairly early and can continue computing new results while previous results are being produced to the user. Thus, response time is the time from the submission of a request until the first response is produced that means time when the task is submitted until the first response is received. So the response time should be low for best scheduling.

So we can conclude that a good scheduling algorithm must possess the following characteristics:

a. Minimum context switches.
b. Maximum CPU utilization.
c. Maximum throughput.
d. Minimum turnaround time.
e. Minimum waiting time.

The aim of CPU scheduling is to assign processes to be executed by the CPU over time, in a way that meets the system objectives, such as minimizing response time, throughput and CPU efficiency (Stallings, 2012).

2.7 Review of Various Scheduling Algorithms

Rakesh *et al.,* (2011), adopted a method by testing the popular algorithms. In the work, they suggested a fit factor 'f' which is computed by:

$$f = UP * UW * SP * BW \quad 2.1$$

where the UP is the User priority set by the designer, UW is the User Priority weight, SP is the Shorter Burst time priority and the BW as the Burst time priority weight. A method which was used as a diligent presentation of pseudo-code was produced for the algorithm. The approach compared the performance of Fittest Job First Dynamic Round Robin (FJFDRR) and the Priority Based Static Round Robin (PBSRR) which they later concluded their approach performs better.

Contribution: The algorithm produced better results when compared to the Priority Based Static Round Robin(PBSRR)

Limitation: computational overhead was high; a lot of time was spent calculating the fit factor formula for each process.

Ali (2012) developed an algorithm for Improving Efficiency of Round Robin Scheduling Using Ascending Quantum and Minumim-Maxumum Burst time. In the Algorithm, incoming processes and their associated burst times are rearranged in ascending order and the quantum time is calculated by multiplying the summation of the minimum and maximum CPU burst with 80 percent. In each round, the procedure is repeated to get the quantum time which is to be used for the remaining processes in the next round and so on. The 80 percent is chosen depending on two reasons: First, if the quantum time calculated depends only on the summation, the algorithm will become SJF. Secondly, the rule of thumb is that 80 percent of the CPU bursts should be shorter than the quantum time.

$$\text{Quantum time} = (\text{MinBurst} + \text{MaxBurst}) * 80\% \quad 2.2$$

Contribution: This algorithm produced better results when compared to the Round Robin by minimizing AWT, ATAT and number of context switching

Limitation: the algorithm did not experience any context switching due to its large quantum time which invalidates the round robin scheduling where some processes are expected to switch to second round

(Simon *et al.*, 2014), proposed an algorithm known as, Half Life variable Quantum Time Round Robin (HLVQTRR). His approach is half of the burst time of each process is executed in the first round and the remaining will be executed in the second round.

Contribution: HLVQTRR minimizes number of context switches and yields better response time when compared to traditional RR.

Limitations: for small BT values, performance metrics would not be optimized by executing only half also, for burst times that are large (over 200ms); executing half may result in starvation of processes. It also did not consider arrival time.

(Simon *et al.*, 2014) developed an algorithm called Dynamic Round Robin with Controlled Preemption. It applies controlled preemption to processes that are using the CPU in RR. If 95 percent of process CPU burst is less than or equal to the Quantum Time, then it should be allowed to run till conclusion otherwise it should be preempted if its time slice expires.

Contribution: DRRCP yields better AWT and ATAT when compared to traditional RR.

Limitation: Average response time was high when compared to traditional RR

(Abdullahi and Junaidu, 2013) developed an algorithm known as Longest Job First with Combinational Burst Time (LJF+CBT) which made an improvement to the Longest Job First (LJF)

CPU scheduling algorithm. This algorithm minimizes the starvation problem in Job First (LJF) CPU scheduling algorithm by minimizing average waiting time, average turnaround time and number of context switches. The algorithm sorts processes in descending order of their burst times and then a threshold known as Combined Weighted Average (Cwa) which is the average of the burst times of processes in the ready queue given by:

$$Cwa = \frac{\sum_{i=1}^{n} bt_i}{n} \quad 2.3$$

Where:

bt_i = burst time of process i

n= number of processes

is defined, which is used to categorize the processes into long and short processes. A process is said to be a Long process if its burst time is greater than Cwa while a process is said to be short if its burst time is less than or equal to Cwa. New burst times are created from this categorization by merging two consecutive shorter processes until no shorter process has one to merge with or no shorter process exist in the categorization. After the merging, new queue is created by sorting the categorized and merged processes in descending order of burst times. The CPU is then allocated to the processes based on Longest Job First. The algorithm produced better results by decreasing AWT and ATAT when compared with the Longest Job First scheduling algorithm. Its limitation however is, by adding a combinational burst time for shorter processes, performance metrics (AWT, ATAT, ART) were still not fully optimized.

2.8 Percentiles

Percentiles are frequently used as indicators of performance in both the academic and corporate worlds. Percentiles provide information about how a person or thing relates to a larger group.

Relative measures of this type are often extremely valuable to researchers employing statistical techniques. The k^{th}*percentile* is a value in a data set that splits the data into two pieces: The lower piece contains *k* percent of the data, and the upper piece contains the rest of the data (which amounts to [100 – *k*] percent, because the total amount of data is 100%). ***Note:****k* is any number between 0 and 100 (Rumsey, 2015).

The median is the 50th percentile: the point in the data where 50% of the data fall below that point, and 50% fall above it.

For example: if a teacher wishes to determine the exam score that divides his class in half, with 50% scoring above and 50% scoring below, he determines the point that marks the 50 percentile.

There is no universal definition of percentile, however all definitions yield similar results when the number of observations is very large (Rumsey,2015).

One definition of percentile, often given in texts, is that the *P*-th percentile ($0 \leq p \leq 100$) of *N* ordered values (arranged from least to greatest) is the smallest value in the data such that Ppercent of the data is less than or equal to that value. This is obtained by first calculating the ordinal rank

$$n = \left\lceil \frac{p}{100} * N \right\rceil \qquad 2.4$$

Where:

n is the position value
P is the percentile
N is number of data values

and then taking the value that corresponds to that rank.

For example: Given these five numbers 15, 20, 35, 40 and 50

the rank of the 30th percentile would be

$$n = \left\lceil \frac{30}{100} * 5 \right\rceil = \lceil 1.5 \rceil = 2 \qquad 2.5$$

Thus the 30th percentile is the second number in the sorted list i.e. 20. The 100th percentile is defined to be the largest value.

CHAPTER THREE

DESIGN OF THE IMPROVED LONGEST JOB FIRST CPU SCHEDULING ALGORITHM

3.1 Proposed Improved Longest Job First CPU Scheduling Algorithms

The proposed algorithm extended the algorithm of Abdullahi and Junaidu (2013) by introducing a new method of:

a. determining the threshold that will be used in the categorization of the processes
b. merging shorter processes

The algorithm arranges all processes in the ready queue in descending order of their burst times. It uses the principle of percentile (equation 3.1) to determine the threshold for categorizing the processes.

$$Q = \left\lceil \left(\frac{percentile}{100} \right) * n \right\rceil \quad (3.1)$$

(Rumsey, 2015)

Where:

Q=Threshold

Percentile=50,60,70,80,90,or100

n = no of processes

The process burst time at the percentile value (from smallest to largest) will be the threshold.

A process is categorized as a long process if its burst time is greater than the threshold otherwise it will be categorized as a short process

$$categorized\ process = \begin{cases} long: bt_i > threshold \\ short: bt_i \leq threshold \end{cases}, \quad i = 1, \dots n$$

At this stage only short processes will be considered. Two shorter processes are merged. This merging will be done by:

a. merging the longest process with the shortest process
b. the second longest process with the shortest-but-one shortest process,
c. the third longest process with the shortest-but-two shortest process
d. and so on until there are no processes left to merge.

The merged processes and the processes in the long category will then be rearranged in descending order of the burst times in the ready queue, and finally, allocate CPU to the processes based on LJF**.**

3.2 Pseudocode of the Proposed Improved Longest Job First CPU Scheduling Algorithm

Let X_i be a process and $\delta_i$be its burst time, temp[]=null, Q= threshold, P= 50,60,70,80,90,100

Algorithm ILJF ($N\left[\left(X_{0,}\delta_0\right),\left(X_{1,}\delta_1\right),\dots\dots,\left(X_{n-1,}\delta_{n-1}\right)\right],P$)

//Input: An array $N\left[\left(X_{0\ ,}\delta_0\right),\left(X_{1\ ,}\delta_1\right),\dots\dots,\left(X_{n-1}\ ,\delta_{n-1}\right)\right]$ of jobs to be assigned to CPU

//Output: An array $N'\left[\left(X_{0\ ,}\delta_0\right),\left(X_{1\ ,}\delta_1\right),\dots\dots,\left(X_{m-1\ ,}\delta_{m-1}\right)\right]$ merged jobs to be assigned to CPU

1. N'=LJFsort(N)
2.$Q = \left\lceil \frac{P}{100} * n \right\rceil$
3.i= N'.length-1
4.j= N'.length-Q
5.while($j \leq i$){
    6.temp.insert(N'[j])

7. N'[j]=null
8.J++
}
9.Min=0
10.Max=temp.length-1
11.While(min<max){
12.t=merge(temp[min], temp[max])
13. N'.insert(t)
14.Min++
15.Max—
}
16.Return N'

3.3 System Architecture

The logical architecture of the system is depicted in Figure 3.1 The system takes in as input the number of processes to be considered, the range (lower and upper bound) of the burst times, the statistical distribution used in generating burst times and the time quantum used by RR CPU scheduling algorithm. The number of processes specified is generated using the specified statistical distribution (i.e. Uniform, Normal or Exponential) for generating the burst times. The generated processes are stored in the process log and are arranged in the ready queue as the scheduling algorithm requires. The CPU is allocated to the processes by using each of the scheduling algorithms in the scheduler module and the performance criteria of each of the algorithms are evaluated and stored in the criteria log. The evaluated criteria are also displayed on the computer screen and the performance graph of each of the criterion for all the algorithms can be drawn.

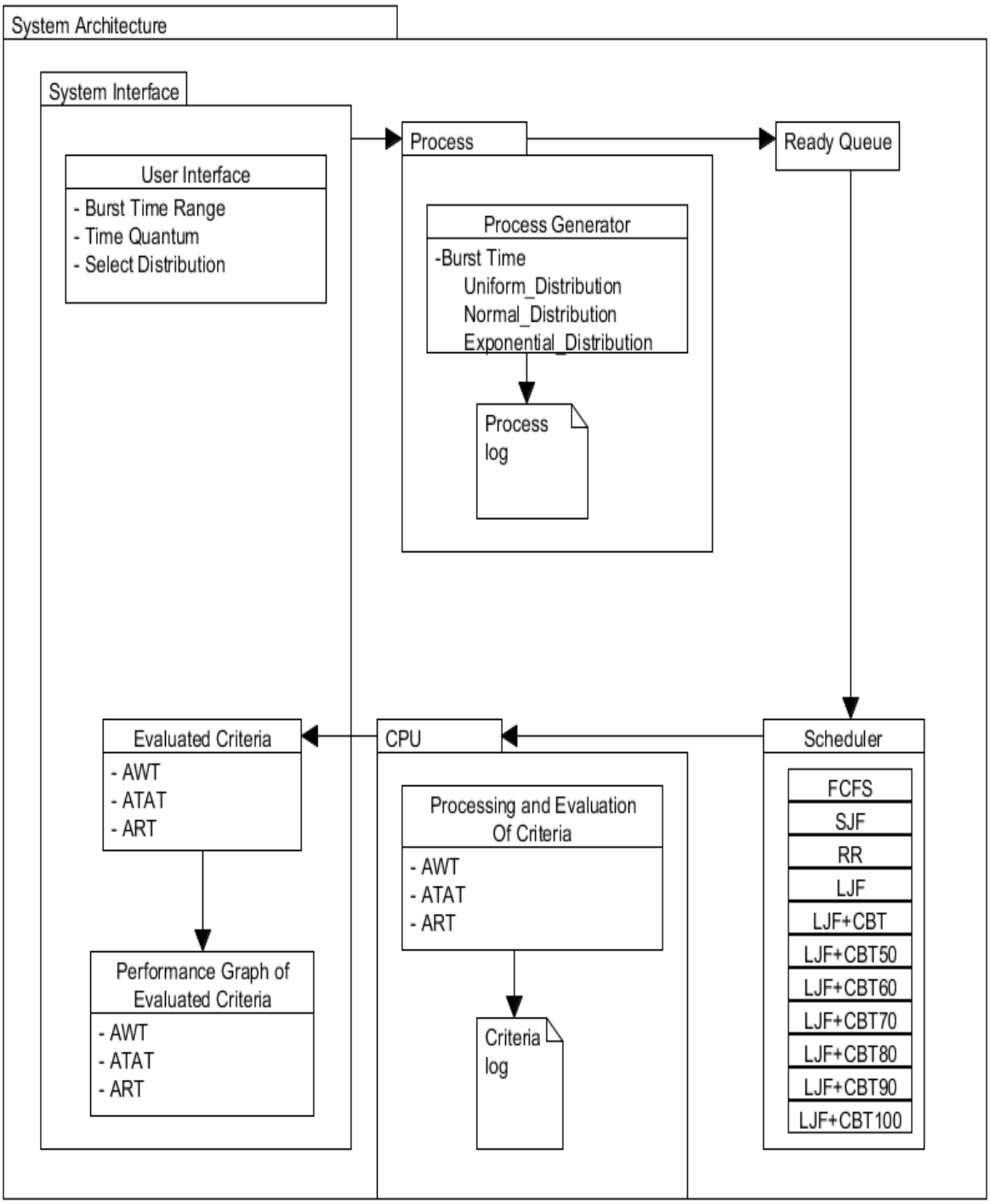


Figure 3. 1: System Architecture of Proposed Algorithm

3.4 Illustrative Example

To describe the proposed algorithm with an example, consider ten processes as shown in Table 3.1. Assuming the arrival time for the given processes is zero (0).

Table 3. 1: Process Table

Process	Burst Time	Arrival time
P1	9	0
P2	27	0
P3	14	0
P4	30	0
P5	22	0
P6	19	0
P7	45	0
P8	39	0
P9	7	0
P10	10	0

Step 1: Processes in Table 3. 1 are arranged in descending order of their burst time as shown in Table 3.2

Table 3. 2: Arranged Processes in Descending Order of Burst Time

Process	Burst time
P7	45
P8	39
P4	30
P2	27
P5	22
P6	19
P3	14
P10	10
P1	9
P9	7

Step 2: The Threshold is determined by calculating the 50th, 60th, 70th, 80th, 90th and 100th percentile value of the burst time of the processes, using the percentile formula in equation (3.1)

$$Q = \left\lceil \frac{p}{100} * n \right\rceil \quad 3.2$$

Where *p* is the percentile value (for this example we use 80th), *n* is the number of processes and *Q* is the threshold

$$Q = \left\lceil \frac{80}{100} * 10 \right\rceil = 8.0 \quad 3.3$$

The burst time of process at the 8th position (from smallest to the largest) will be used as the threshold. Thus in Table 3. 2, the process at the 8th position is P4 with burst time 30. Therefore, 30 will be used as the threshold.

Step 3: Categorization

a. All processes greater than 30 are considered long, these include: P8 and P7 with burst time 39 and 45 respectively.

b. All processes less than or equal to 30 are considered short, these include: P4, P2, P5, P6, P3, P10, P1 and P9 with burst times 30, 27, 22, 19, 14, 10, 9 and 7 respectively.

Step 4: Merging of processes in short category

a. When two processes are merged, a new identity is formed by combining the identities of the two merged processes. For example: process P1 merged with process P2 give a new identity as *P1,2. Where the '*,' implies that it is a merged process and the numbers represent the original identity numbers of the processes separated by a comma (,).

From Table 3.2 the merging of processes in short category is done by merging the longest process with the shortest process, which gives:

*P4,9=30+7=37

Then second longest process with shortest but One Process, which gives:

*P2,1=36

Third longest process with shortest but Two Process, which gives:

*P5,10=32

Finally, fourth longest process with shortest but Three Process, which gives:

*P6,3=33

The results of the merged processes in the short category are shown in Table 3.3

Table 3. 3: Merged Processes

Process	Burst Time
*P4,9	37
*P2,1	36
*P5,10	32
*P6,3	33

The new process table for the all the processes is shown in Table 3. 4

Table 3. 4: The New Process Table

Process	Burst Time
P7	45
P8	39
*P4,9	37
*P2,1	36
*P5,10	32
*P6,3	33

Step 5: Rearranging the processes in Table 3. 4 in descending order of burst time is shown in Table 3. 5

Table 3. 5: Burst Time of Rearranged Processes in Descending Order

Process	Burst Time
P7	45
P8	39
*P4,9	37
*P2,1	36
*P6,3	33
*P5,10	32

Step 6: CPU is assigned to processes in LJF format, as represented in the Gantt chart shown in Figure 3.2.

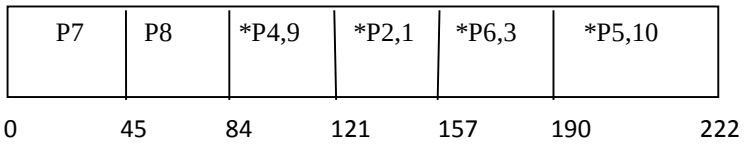


Figure 3. 2: Gantt Chart for Proposed Algorithm

3.5 Evaluation of FCFS, SJF, RR, LJF, LJF+CBT and the modified LJF+CBT50, LJF+CBT60, LJF+CBT70, LJF+CBT80, LJF+CBT90, LJF+CBT100

ASSUMPTIONS

Three performance criteria- average waiting time, average turnaround time, and average response time were studied. To observe these criteria, the following CPU scheduling algorithms were simulated:

a. First Come First Serve (FCFS)
b. Shortest Job First (SJF)
c. Round Robin (RR)
d. Longest Job First (LJF)
e. Longest Job First with Combinational Burst Time (LJF+CBT)
f. The modified Longest Job First with Combinational Burst Time (LJF+CBT 50)
g. The modified Longest Job First with Combinational Burst Time (LJF+CBT 60)
h. The modified Longest Job First with Combinational Burst Time (LJF+CBT 70)
i. The modified Longest Job First with Combinational Burst Time (LJF+CBT 80)
j. The modified Longest Job First with Combinational Burst Time (LJF+CBT 90)
k. The modified Longest Job First with Combinational Burst Time (LJF+CBT 100).

The simulation environment was a single processor environment and all the processes are independent and CPU bound, no process was I/O bound. The system was also assumed to have no context switch cost i.e. the context switching time is equal to zero which means there was no context switch overhead incurred in transferring from one process to another.

3.6 Experimental Analysis

To demonstrate the previous considerations, the following example in Table 3.6 shows the burst times of ten processes. These burst times are used to evaluate and compare the eleven CPU scheduling algorithms. All processes are assumed to be in the ready queue and their arrival times considered to be zero. The time quantum used by RR is 10ms as was defined in (Abdullahi and Junaidu, 2013) for easy comparison.

Table 3. 6: Process Table

Process	Burst Time	Arrival time
P1	9	0
P2	27	0
P3	14	0
P4	30	0
P5	22	0
P6	19	0
P7	45	0
P8	39	0
P9	7	0
P10	10	0

For evaluation purpose, the formula of Waiting Time (WT) (i.e. (equation 3.4)) is used in calculating the Average Waiting Time (AWT) (i.e. (equation 3.5)) for each scheduling algorithm.

Waiting Time (WT) = Time first Scheduled – Arrival time 3.4

Average Waiting Time (AWT) = Sum of all Processes Waiting Time/ Total numbers of Processes. for each process 3.5

The Turnaround Time (TAT) (i.e. (equation 3.6)) is used in calculating the Average Turnaround Time (ATAT) (i.e. (equation 3.7)) for each scheduling algorithm.

Turn Around Time (TAT) = Time of Process completion – Arrival Time 3.6

Average Turn-Around Time = Sum of each Process / Total numbers of Processes. 3.7

And the Response Time (RT) (i.e. equation 3.8) is used in calculating the Average Response Time (ART) (i.e. equation 3.9) for each scheduling algorithm.

Response Time (RT) = Time of process's first Response – Arrival time 3.8

Average Response Time (RT) = Sum of all Processes Response Time/ Total numbers of Processes. *3.9*

3.6.1 First Come First Serve (FCFS)

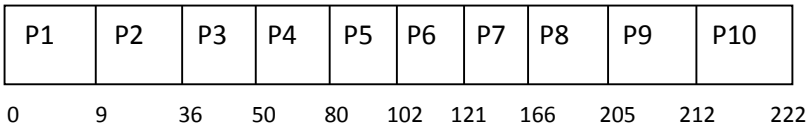


Figure 3. 3 : Gantt Chart of First Come First Serve

Waiting Time

P1: (0-0) = 0, P2: (9-0) = 9, P3: (36-0) = 36, P4: (50-0) = 50, P5: (80-0) = 80, P6: (102-0) = 102, P7: (121-0) = 121, P8: (166-0) = 166, P9: (205-0) = 205, and P10: (212-0) = 212.

Average Waiting Time

$$AWT = \frac{(0 + 9 + 36 + 50 + 80 + 102 + 121 + 166 + 205 + 212)}{10} = \frac{981}{10} = 98.1$$

Turnaround Time

P1: (9-0 = 9, P2: (36-0) =36, P3: (50-0) = 50, P4: (80-0) =80, P5: (102-0) = 102, P6: (121-0) = 121, P7: 166-0) = 166, P8: (205-0) = 205, P9: (212-0) = 212, P10: (222-0) = 222.

Average Turnaround Time

$$\text{ATAT} = \frac{9 + 36 + 50 + 80 + 102 + 121 + 166 + 205 + 212 + 222}{10} = \frac{1203}{10}$$

$$= 120.3$$

Response Time

P1: (0-0) = 0, P2: (9-0) = 9, P3: (36-0) = 36, P4: (50-0) = 50, P5: (80-0) = 80, P6: (102-0) = 102, P7: (121-0) = 121, P8: (166-0) = 166, P9: (205-0) = 205, and P10: (212-0) = 212.

Average Response Time

$$\text{ART} = \frac{0 + 9 + 36 + 50 + 80 + 102 + 121 + 166 + 205 + 212}{10} = \frac{981}{10} = 98.1$$

3.6.2 Shortest Job First (SJF)

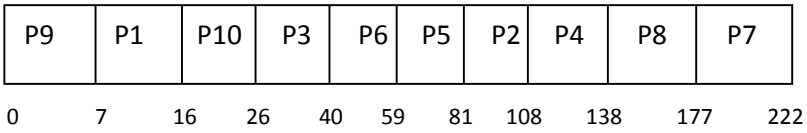


Figure 3. 4: Gantt Chart of Shortest of Job First

Waiting Time

P1: (7-0) = 7, P2: (81-0) = 81, P3: (26-0) = 26, P4: (108-0) = 108, P5: (59-0) = 59, P6: (40-0) = 40, P7: (177-0) = 177, P8: (138-0) = 138, P9: (0-0) = 0, and P10: (16-0) = 16.

Average Waiting Time

$$\text{AWT} = \frac{(7 + 81 + 26 + 108 + 59 + 40 + 177 + 138 + 0 + 16)}{10} = \frac{652}{10} = 65.2$$

Turnaround Time

P1: (16-0) = 16, P2: (108-0) =108, P3: (40-0) = 40, P4: (138-0) =138, P5: (81-0) = 81, P6: (59-0) = 59, P7: (222-0) = 222, P8: (177-0) = 177, P9: (7-0) = 7, P10: (26-0) = 26.

Average Turnaround Time

$$\text{ATAT} = \frac{16 + 108 + 40 + 138 + 81 + 59 + 222 + 177 + 7 + 26}{10} = \frac{874}{10} = 87.4$$

Response Time

P1: (7-0) = 7, P2: (81-0) = 81, P3: (26-0) = 26, P4: (108-0) = 108, P5: (59-0) = 59, P6: (40-0) = 40, P7: (177-0) = 177, P8: (138-0) = 138, P9: (0-0) = 0, and P10: (16-0) = 16.

Average Response Time

$$\text{ART} = \frac{7 + 81 + 26 + 108 + 59 + 40 + 177 + 138 + 0 + 16}{10} = \frac{652}{10} = 65.2$$

3.6.3 Longest Job First (LJF)

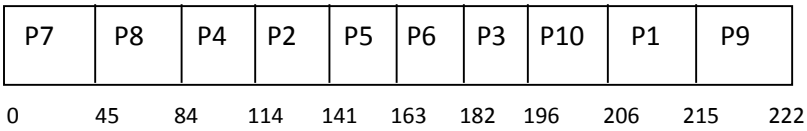


Figure 3. 5: Gantt Chart of Longest Job First

Waiting Time

P1: (206-0) = 206, P2: (114-0) = 114, P3: (182-0) = 182, P4: (84-0) = 84, P5: (141-0) = 141, P6: (163-0) =163, P7: (0-0) = 0, P8: (45-0) = 45, P9: (215-0) = 0, and P10: (196-0) = 196.

Average Waiting Time

$$\text{AWT} = \frac{(206 + 114 + 182 + 84 + 141 + 163 + 0 + 45 + 215 + 196)}{10} = \frac{1301}{10}$$

$$= 130.1$$

Turnaround Time

P1: (215-0) = 215, P2: (141-0) =141, P3: (196-0) =196, P4: (114-0) =114, P5: (163-0) = 163, P6: (182-0) = 182, P7: (45-0) = 45, P8: (84-0) = 84, P9: (222-0) =222, P10: (206-0) = 206.

Average Turnaround Time

$$\text{ATAT} = \frac{215 + 141 + 196 + 114 + 163 + 182 + 45 + 84 + 222 + 206}{10} = \frac{1568}{10}$$

$$= 156.8$$

Response Time

P1: (206-0) = 206, P2: (114-0) = 114, P3: (182-0) = 182, P4: (84-0) = 84, P5: (141-0) = 141, P6: (163-0) =163, P7: (0-0) = 0, P8: (45-0) = 45, P9: (215-0) = 0, and P10: (196-0) = 196.

Average Response Time

$$\textbf{ART} = \frac{206 + 114 + 182 + 84 + 141 + 163 + 0 + 45 + 215 + 196}{10} = \frac{1301}{10}$$

$$= 130.1$$

3.6.4 Round Robin (RR)

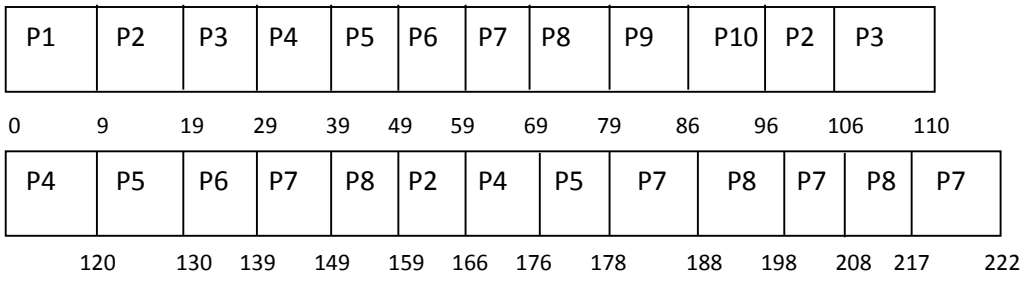


Figure 3. 6: Gantt Chart of Round Robin

Waiting Time

P1: (0-0) = 0, P2: ((9+77+53)-0) =139, P3: ((19+77)-0) = 96, P4: ((29+71+46)-0) = 146, P5: ((39+71+46)-0) = 156, P6: ((49+71)-0) = 120, P7: ((59+70+29+10+9)-0) = 177, P8: ((69+70+29+10)-0) = 178, P9: ((79)-0) = 79, P10: (86-0) = 86.

Average Waiting Time

$$\text{AWT} = \frac{(0 + 139 + 96 + 146 + 156 + 120 + 177 + 178 + 79 + 86)}{10} = \frac{1177}{10}$$

$$= 117.7$$

Turnaround Time

P1: (9-0) = 9, P2: (166-0) =166, P3: (110-0) = 110, P4: (176-0) =176, P5: (178-0) = 178, P6: (139-0) = 139, P7: (222-0) = 222, P8: (217-0) = 217, P9: (86-0) = 86, P10: (96-0) = 96.

Average Turnaround Time

$$\text{ATAT} = \frac{9 + 166 + 110 + 176 + 178 + 139 + 222 + 217 + 86 + 96}{10} = \frac{1399}{10}$$

$$= 139.9$$

Response Time

P1: (0-0) = 0, P2: (9-0) = 9, P3: (19-0) = 19, P4: (29-0) = 29, P5: (39-0) = 39, P6: (49-0) = 49, P7: (59-0) = 59, P8: (69-0) = 69, P9: (79-0) = 79, and P10: (86-0) = 86.

Average Response Time

$$\text{ART} = \frac{0 + 9 + 19 + 29 + 39 + 49 + 59 + 69 + 79 + 86}{10} = \frac{438}{10} = 43.8$$

3.6.5 Longest Job First with Combinational Burst Time (LJF+CBT)

The LJF+ CBT algorithm first sorted the given process in descending order of their burst time, (Table 3.7) then merged the new processes and finally arranged them in descending order of their burst time (Table 3.8)

Table 3. 7: New Identification and Burst Time of Processes

Process	Burst Time
P7	45
P8	39
P4	30
P2	27
P5	22
P6	19
P3	14
P10	10
P1	9
P9	7

Table 3. 8 New Identification and Burst Time of Processes

Process	**Burst Time**
P7	45
*P5,6	41
P8	39
P4	30
P2	27
*P3,10	24
*P9,1	16

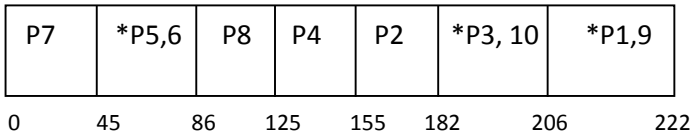


Figure 3. 7: Gantt Chart of LJF+CBT

Waiting Time

*P1, 9: (206-0) = 206, where *P1, 9 is the result of merging P1 and P9, at Time = 0 and thus had to wait for the longer processes to proceed. P2: (155-0) = 155, where P2 arrived at Time = 0 and scheduled at 155. *P3, 10: (182-0) = 182, P4: (125-0) = 125, *P5, 6: (45-0) = 45, P7: (0-0) = 0, P8: (86-0) = 86

Average Waiting Time

$$\text{AWT} = \frac{(206 + 155 + 182 + 125 + 45 + 0 + 86)}{10} = \frac{799}{10} = 79.9$$

Turnaround Time

*P1, 9: (222-0) = 222, P2: (182-0) =182, *P3, 10: (206-0) =206, P4: (155-0) =155, *P5, 6: (86-0) = 86, P7: (45-0) = 45, P8: (125-0) = 125.

Average Turnaround Time

$$\text{ATAT} = \frac{(222 + 182 + 206 + 155 + 86 + 45 + 125)}{10} = \frac{1021}{10} = 102.1$$

Response Time

*P1, 9: (206-0) = 206, P2: (155-0) = 155, *P3, 10: (182-0) = 182, P4: (125-0) = 125, *P5, 6: (450) = 45, P7: (0-0) = 0, P8: (86-0) = 86

Average Response Time

$$\text{ART} = \frac{206 + 155 + 182 + 125 + 45 + 0 + 86}{10} = \frac{799}{10} = 79.9$$

3.6.6 Longest Job First with Combinational Burst Time (LJF+CBT 50)

Table 3. 9: New Identification and Burst Time of Processes

PROCESS	BURST TIME
P7	45
P8	39
P4	30
P2	27
*P6,9	26
*P3,1	23
P5	22
P10	10

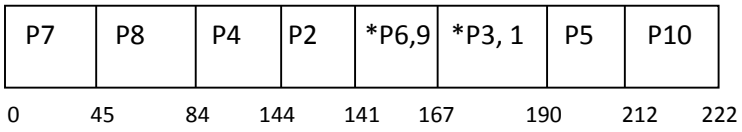


Figure 3. 8: Gantt Chart of LJF+CBT50

Waiting Time

P2: (114-0) = 114, where P2 arrived at Time = 0 and scheduled at 144., *P3, 1: (167-0) = 167, P4: (84-0) = 84, P5: (190-0) = 190, *P6,9: (141-0) =141, P7: (0-0) = 0, P8: (45-0) = 45, P10: (212-0) = 212

Average Waiting Time

$$\text{AWT} = \frac{(114 + 167 + 84 + 190 + 141 + 0 + 45 + 212)}{10} = \frac{983}{10} = 98.3$$

Turnaround Time

P2: (141-0) = 141, *P3, 1: (190-0) = 190, P4: (114-0) =114, P5: (212-0) = 212, *P6,9: (167-0) =167, P7: (45-0) =45, P8: (84-0) = 84, P10: (222-0) = 222

Average Turnaround Time

$$\text{ATAT} = \frac{(141 + 190 + 114 + 212 + 167 + 45 + 84 + 222)}{10} = \frac{1175}{10} = 117.5$$

Response Time

P2: (114-0) = 114, *P3, 1: (167-0) = 167, P4: (84-0) = 84, P5: (190-0) = 190, *P6,9: (141-0) =141, P7: (0-0) = 0, P8: (45-0) = 45, P10: (212-0) = 212

Average Response Time

$$\textbf{ART} = \frac{114 + 167 + 84 + 190 + 141 + 0 + 45 + 212)}{10} = \frac{983}{10} = 98.3$$

3.6.7 Longest Job First with Combinational Burst Time (LJF+CBT 60)

Table 3. 10 New Identification and Burst Time of Processes

PROCESS	BURST TIME
P7	45
P8	39
P4	30
*P5, 9	29
*P6, 1	28
P2	27
*P3,10	24

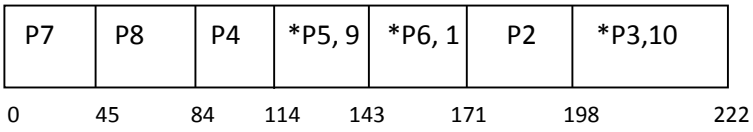


Figure 3. 9: Gantt Chart of LJF+CBT60

Waiting Time

P2: (171-0) = 171, *P3, 10: (198-0) = 198, P4: (84-0) = 84, *P5, 9: (114-0) = 114, *P6,1: (143-0) =143, P7: (0-0) = 0, P8: (45-0) = 45.

Average Waiting Time

$$\text{AWT} = \frac{(171 + 198 + 84 + 114 + 143 + 0 + 45)}{10} = \frac{755}{10} = 75.5$$

Turnaround Time

P2: (198-0) = 198, *P3, 10: (222-0) = 222, P4: (114-0) = 114, *P5, 9: (143-0) = 143, *P6,1: (171-0) =171, P7: (45-0) = 45, P8: (84-0) = 84.

Average Turnaround Time

$$\text{AWT} = \frac{(198 + 222 + 114 + 143 + 171 + 45 + 84\ )}{10} = \frac{977}{10} = 97.7$$

Response Time

P2: (171-0) = 171, *P3, 10: (198-0) = 198, P4: (84-0) = 84, *P5, 9: (114-0) = 114, *P6,1: (143-0) =143, P7: (0-0) = 0, P8: (45-0) = 45.

Average Response Time

$$\text{ART} = \frac{171 + 198 + 84 + 114 + 143 + 0 + 45}{10} = \frac{755}{10} = 75.5$$

3.6.8 Longest Job First with Combinational Burst Time (LJF+CBT 70)

Table 3. 11 New Identification and Burst Time of Processes

PROCESS	BURST TIME
P7	45
P8	39
*P2, 9	34
*P5, 1	31
P4	30
*P6, 10	29
P3	14

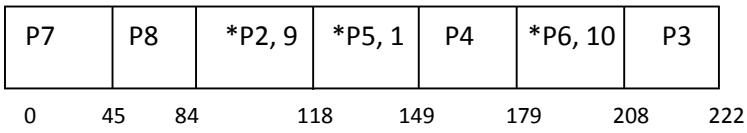


Figure 3. 10: Gantt Chart of LJF+CBT70

Waiting Time

*P2,9: (84-0) = 84, P3: (208-0) =208, P4: (149-0) = 149, *P5, 1: (118-0) = 118, *P6,10: (179-0) =179, P7: (0-0) = 0, P8: (45-0) = 45.

Average Waiting Time

$$\text{AWT} = \frac{(84 + 208 + 149 + 118 + 179 + 0 + 45+)}{10} = \frac{783}{10} = 77.8$$

Turnaround Time

*P2,9: (118-0) = 118, P3: (222-0) =222, P4: (179-0) = 179, *P5, 1: (149-0) = 149, *P6,10: (208-0) =208, P7: (45-0) =45, P8: (84-0) =8 4.

Average Turnaround Time

$$\text{ATAT} = \frac{(118 + 222 + 179 + 149 + 208 + 45 + 84)}{10} = \frac{1000}{10} = 100$$

Response Time

*P2,9: (84-0) = 84, P3: (208-0) =208, P4: (149-0) = 149, *P5, 1: (118-0) = 118, *P6,10: (179-0) =179, P7: (0-0) = 0, P8: (45-0) = 45.

Average Response Time

$$\text{ART} = \frac{84 + 208 + 149 + 118 + 179 + 0 + 45}{10} = \frac{783}{10} = 77.8$$

3.6.9 Longest Job First with Combinational Burst Time (LJF+CBT 80)

Table 3. 12 New Identification and Burst Time of Processes

PROCESS	BURST TIME
P7	45
P8	39
*P4, 9	37
*P2, 1	36
*P6,3	33
*P5,10	32

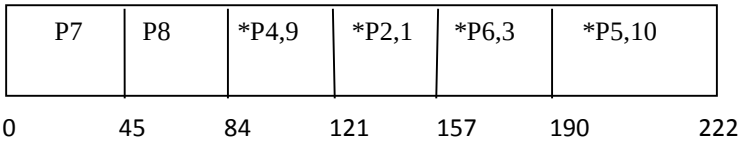


Figure 3. 11: Gantt Chart of LJF+CBT80

Waiting Time

*P2,1: (121-0) = 121, *P4, 9 (84-0) =84, * P5, 10: (190-0) = 190, *P6, 3: (157-0) = 157, P7: (0-0) =0, P8: (45-0) = 45.

Average Waiting Time

$$\text{AWT} = \frac{(121 + 84 + 190 + 157 + 0 + 45)}{10} = \frac{597}{10} = 59.7$$

Turnaround Time

*P2,1: (157-0) = 157, *P4, 9 (121-0) =121, * P5, 10: (222-0) = 222, *P6, 3: (190-0) = 190, P7: (45-0) =45, P8: (84-0) = 84.

Average Turnaround Time

$$\text{ATAT} = \frac{(157 + 121 + 222 + 190 + 45 + 84)}{10} = \frac{819}{10} = 81.9$$

Response Time

*P2,1: (121-0) = 121, *P4, 9 (84-0) =84, * P5, 10: (190-0) = 190, *P6, 3: (157-0) = 157, P7: (0-0) =0, P8: (45-0) = 45.

Average Response Time

$$\text{ART} = \frac{121 + 84 + 190 + 157 + 0 + 45}{10} = \frac{59.7}{10} = 59.7$$

3.6.10 Longest Job First with Combinational Burst Time (LJF+CBT 90)

Table 3. 13 New Identification and Burst Time of Processes

Process	Burst Time
*P8,9	46
P7	45
*P4,1	39
*P2,10	37
*P5,3	36
P5	22

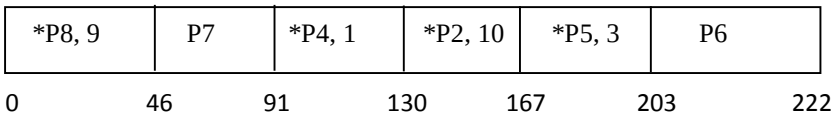


Figure 3. 12: Gantt Chart of LJF+CBT90

Waiting Time

*P2,10: (130-0) = 130, *P4, 1: (91-0) =91, * P5, 3: (167-0) = 167, P6: (203-0) = 203, P7: (46-0) =46, *P8, 9: (0-0) = 0.

Average Waiting Time

$$\text{AWT} = \frac{(130 + 91 + 167 + 203 + 46 + 0)}{10} = \frac{637}{10} = 63.7$$

Turnaround Time

*P2,10: (167-0) = 167, *P4, 1: (130-0) =130, * P5, 3: (203-0) = 203, P6: (222-0) = 222, P7: (91-0) =91, *P8, 9: (46-0) = 46.

Average Turnaround Time

$$\text{ATAT} = \frac{(167 + 130 + 203 + 222 + 91 + 46)}{10} = \frac{859}{10} = 85.9$$

Response Time

*P2,10: (130-0) = 130, *P4, 1: (91-0) =91, * P5, 3: (167-0) = 167, P6: (203-0) = 203, P7: (46-0) =46, *P8, 9: (0-0) = 0.

Average Response Time

$$\text{ART} = \frac{130 + 91 + 167 + 203 + 46 + 0}{10} = \frac{637}{10} = 63.7$$

3.6.11 Longest Job First with Combinational Burst Time (LJF+CBT 100)

Table 3. 14: New Identification and Burst Time of Processes

Process	Burst Time
*P7,9	52
*P8,1	48
*P4,10	40
*P2,3	41
*P5,6	41

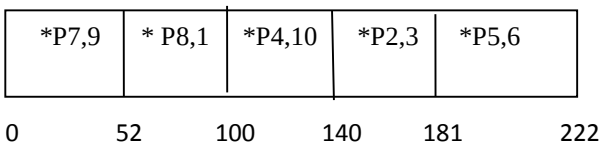


Figure 3. 13**:** Gantt Chart of LJF+CBT100

Waiting Time

*P2,3: (140-0) = 140, *P4, 10: (100-0) =100, * P5, 6: (181-0) = 181, *P7,9: (0-0) = 0, *P8, 1: (52-0) =52.

Average Waiting Time

$$\text{AWT} = \frac{(140 + 100 + 181 + 0 + 52)}{10} = \frac{473}{10} = 47.3$$

Turnaround Time

*P2,3: (181-0) = 181, *P4,10: (140-0) =140, * P5,6: (222-0) = 222, *P7,9: (52-0) = 52, *P8, 1: (100-0) =100.

Average Turnaround Time

$$\text{ATAT} = \frac{(181 + 140 + 222 + 52 + 100)}{10} = \frac{695}{10} = 69.5$$

Response Time

*P2,3: (140-0) = 140, *P4, 10: (100-0) =100, * P5, 6: (181-0) = 181, *P7,9: (0-0) = 0, *P8, 1: (52-0) =52.

Average Response Time

$$\text{ART} = \frac{(140 + 100 + 181 + 0 + 52)}{10} = \frac{473}{10} = 47.3$$

Table 3. 15: Comparative Results of the Algorithms

Algorithms	**Average Waiting Time**	**Average Turn Around Time**	**Average Response Time**
FCFS	98.1	120.3	98.1
SJF	65.2	87.4	65.2
RR	117.7	139.9	43.8
LJF	135.6	156.8	135.6
LJF+CBT	79.9	102.1	79.9
LJF+CBT (50)	98.3	117.5	98.3
LJF+CBT (60)	75.5	97.7	75.5
LJF+CBT (70)	77.8	100	77.8
LJF+CBT (80)	59.7	81.9	59.7
LJF+CBT (90)	63.7	85.9	63.7
LJF+CBT (100)	47.3	69.5	47.3

Table 3.15 shows the comparative result of the algorithms under study. The proposed algorithm with 100^{th} percentile produced best results in terms of AWT and ATAT. This is because a higher percentile value allows for more merging of processes. The algorithm at 90^{th} percentile came second, followed by 80^{th}percentile, SJF, 60^{th} percentile, 70^{th} percentile, LJF+CBT, 50^{th} percentile, FCFS, LJF and RR respectively.

RR Produced best result in terms of ART because the shorter the time slice, the better the performance of RR under the response-time metric.

.

CHAPTER FOUR

IMPLEMENTATION OF THE MODIFIED IMPROVED LONGEST JOB FIRST CPU SCHEDULING ALGORITHM

4.1 Introduction

In chapter three, the eleven algorithms were analyzed using analytic technique with just ten processes. But in order to understand the behavior of these algorithms and draw valid conclusions, there was the need to develop a simulator that considers a wide range of datasets. Using uniform, normal and exponential distribution functions, a simulator was designed to generate random variables which were used as processes and their associated CPU bursts to compute their average waiting time, average turnaround time and average response time. The results were presented in graphical form for proper analysis.

4.2 Description of Simulation

The simulator is designed using Java programming language. It uses uniform, normal or exponential distribution respectively to generate random processes and their associated CPU bursts. The input parameters that the simulator uses are: quantum time (QT) (to be used by the traditional RR), burst time range and number of process. The mean (μ), standard deviation (σ) and exponential distribution parameter (λ) used by the various distributions considered are derived from the burst times. By running the simulator, it will automatically generate the processes and their associated CPU burst which all the scheduling algorithms under consideration will use to test for their respective performance metrics. Finally, it computes the average waiting time, average turnaround time and average response time for the algorithms under consideration.

4.3 Pseudocode for Simulator

Begin
Initialize the Simulator
//N=number of processes, QT= Quantum Time & BT= Burst Time Range
Enter QT
Enter N
Enter BT range
If (Uniform OR Normal OR Exponential)
    Generate the Processes associated CPU burst time
    Compute result for each scheduling Algorithms
//The results are: Average Waiting Time, Average Turn Around Time, Average Response Time
Clear table
End

4.4 System specification

The hardware and software specification used for the simulation of the algorithms are as follows

Hardware

a Hewlett Packard (HP) laptop with a (Tm)i3-311om processor running at 2.40GHz

b 4.00GB of RAM and

c 500GB of hard disk

Software

a Windows 8 operating system

b NetBeans IDE 7. 3.1 version and JDK1.7

A process generator routine was built to generate the process sets. Each process in the process set is a tuple: < (process_*id*, *CPU_time*)>.

A process burst time generator was developed to take care of the random burst time of different processes in the system.

Burst time (i.e. the *CPU_time*) was generated using normal, uniform or exponential distribution.

Normal distribution: a random variable X with mean $-\infty<\mu<\infty$, variance $\sigma^2 >0$ and Standard deviation σ has a normal distribution if it has the pdf

$$f(x)= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad 4.1$$

(Oladugba *et al*., 2014)

Uniform distribution: Consider a random variable X that is uniformly distributed on the interval

[a, b]. So, using the Inverse Transformation method:

The probability density function (pdf) is given by:

$$f(x) = \frac{1}{b-a} \quad 4.2$$

Where $a \leq x \leq b$

And the cumulative density function is given by:

$$f(x) = \frac{x-a}{b-a} = R \quad \text{or} \quad x = a + (b-a)R \quad 4.3$$

Which is a reasonable guess for generating x and R is always a random number on [0, 1] (Jerry *et al.*, 2005).

Its mean and variance respectively are given by:

$$E(X) = \frac{a+b}{2} \; and \; V(X) = \frac{(b-a)^2}{12} \quad 4.4$$

Exponential distribution: Consider a random variable X that is exponentially distributed with a parameter rate of exponential distribution λ. The probability density function (pdf) is given by:

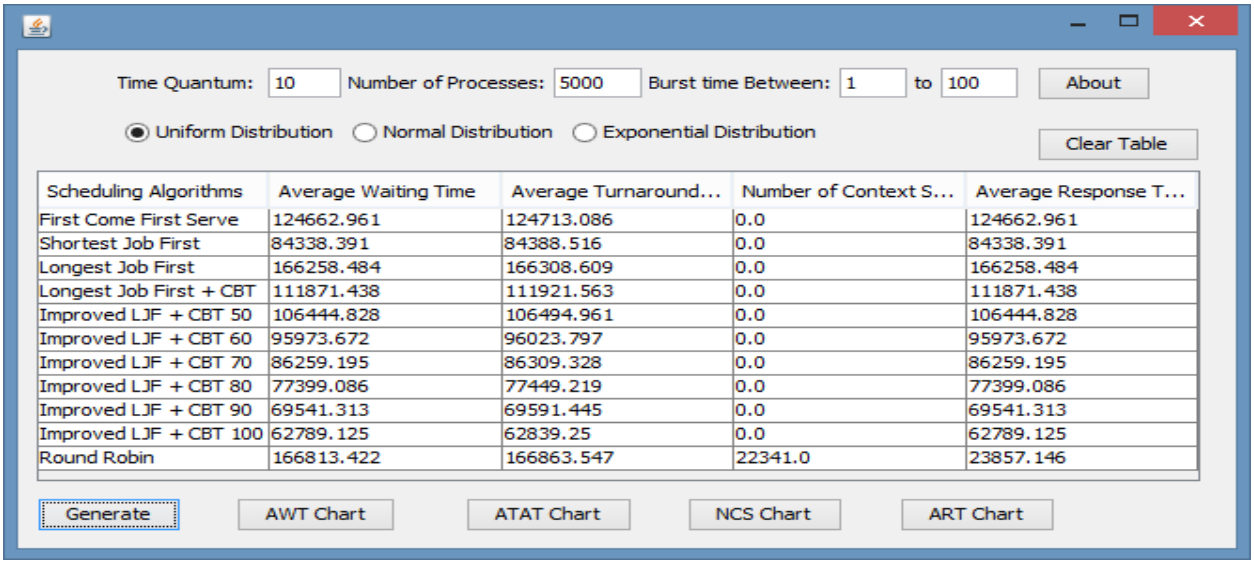
$$f(x) = \lambda e^{-\lambda x} \quad 4.5$$

It has mean and variance respectively are given by:

$$E(X) = \frac{1}{\lambda} and V(X) = \frac{1}{\lambda^2} \quad 4.6$$

4.5 Discussion of Results

Figure 4.1 shows the system interface which presented some results computation using 5000 processes generated by the uniform distribution with burst times ranging between 1ms and 100ms, and the time quantum of 10ms for RR



Scheduling Algorithms	Average Waiting Time	Average Turnaround...	Number of Context S...	Average Response T...
First Come First Serve	124662.961	124713.086	0.0	124662.961
Shortest Job First	84338.391	84388.516	0.0	84338.391
Longest Job First	166258.484	166308.609	0.0	166258.484
Longest Job First + CBT	111871.438	111921.563	0.0	111871.438
Improved LJF + CBT 50	106444.828	106494.961	0.0	106444.828
Improved LJF + CBT 60	95973.672	96023.797	0.0	95973.672
Improved LJF + CBT 70	86259.195	86309.328	0.0	86259.195
Improved LJF + CBT 80	77399.086	77449.219	0.0	77399.086
Improved LJF + CBT 90	69541.313	69591.445	0.0	69541.313
Improved LJF + CBT 100	62789.125	62839.25	0.0	62789.125
Round Robin	166813.422	166863.547	22341.0	23857.146

Figure 4. 1: System Interface for 5000 Processes Using Uniform Distribution

Figure 4.2 shows the graphical result of AWT of 5000 processes obtained in figure 4.1. It was observed that the proposed algorithm with 100thpercentile was better compared to others. This is because it produced lowest results i.e., 62789.125ms, followed by 90th percentile, then 80th percentile SJF, 60th percentile,50th percentile, LJF+CBT, FCFS, LJF and RR respectively.

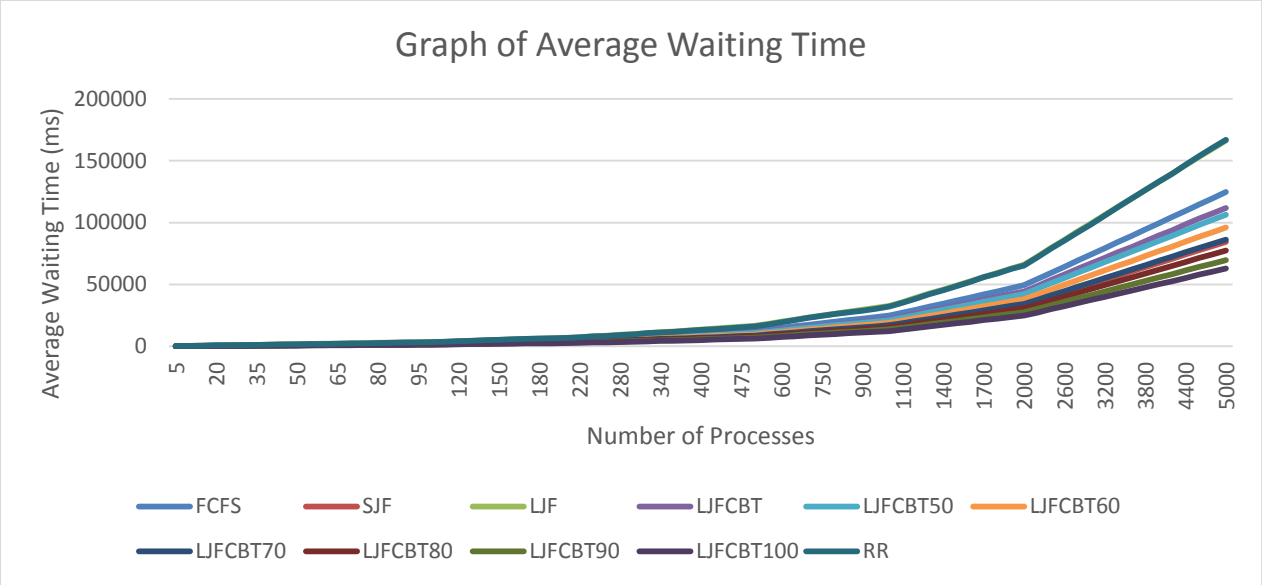


Figure 4. 2: Graph of AWT for 5000 Processes Using Uniform Distribution

Figure 4.3 shows the graphical result of ATAT of 5000 processes obtained in figure 4.1. It was observed that the proposed algorithm with 100thpercentile was better compared to others. This is because it produced lowest results i.e., 62839.25ms, followed by 90th percentile, then 80th percentile SJF, 60th percentile,50th percentile, LJF+CBT, FCFS, LJF and RR respectively.

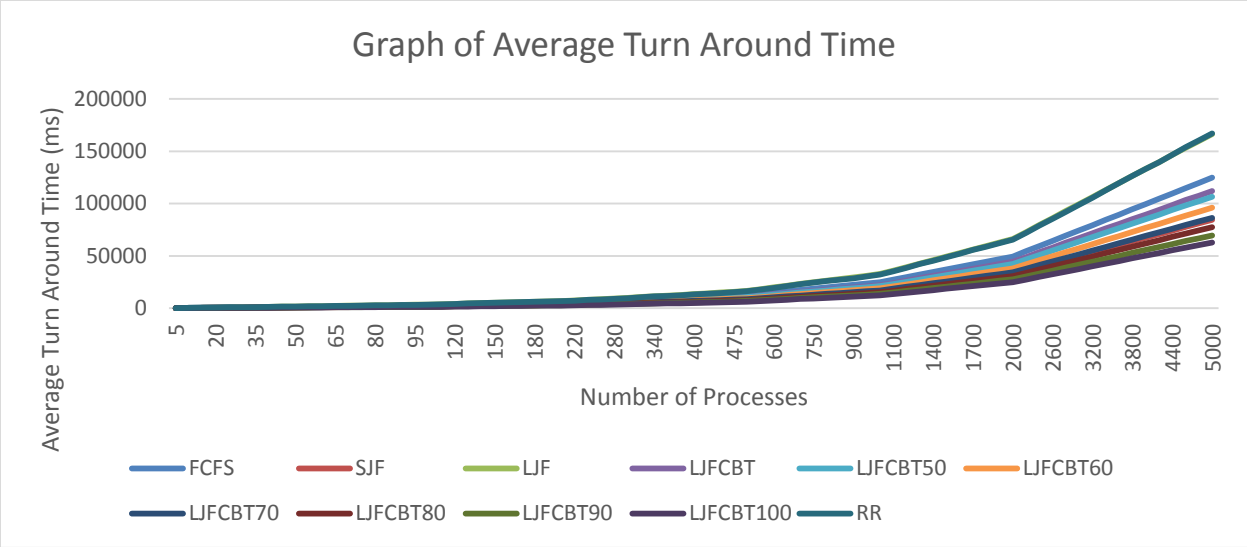


Figure 4. 3: Graph of ATAT for 5000 Processes Using Uniform Distribution

Figure 4.4 shows the graphical result of ART of 5000 processes obtained in figure 4.1. It was observed that the RR was better compared to others. This is because it produced lowest results i.e., 23857.146ms, followed by the proposed algorithm with 100thpercentile then 90th percentile,80th percentile SJF, 60th percentile,50th percentile, LJF+CBT, FCFS and LJF respectively.

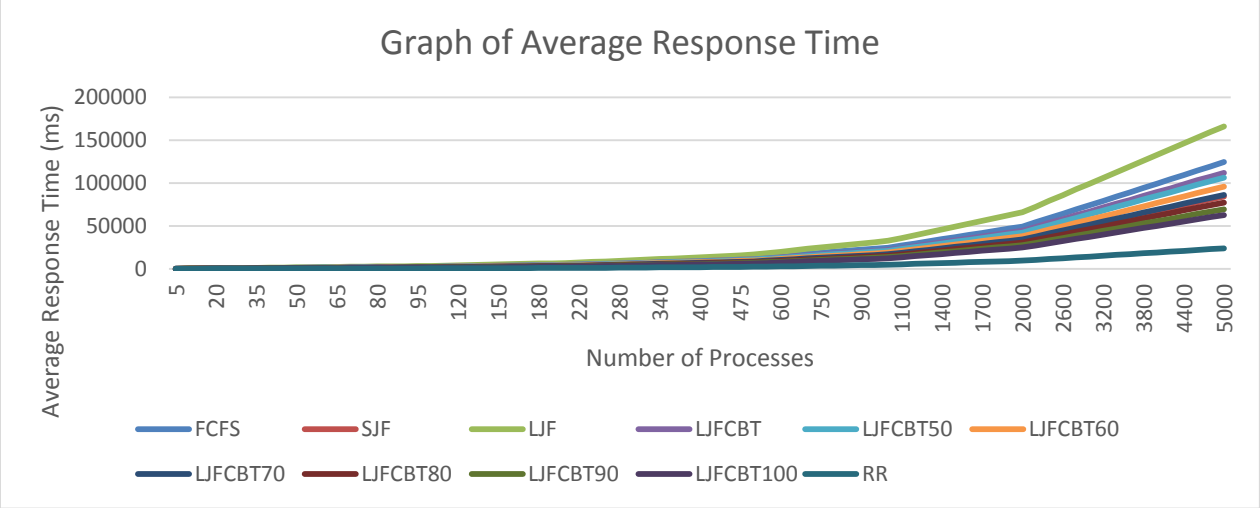
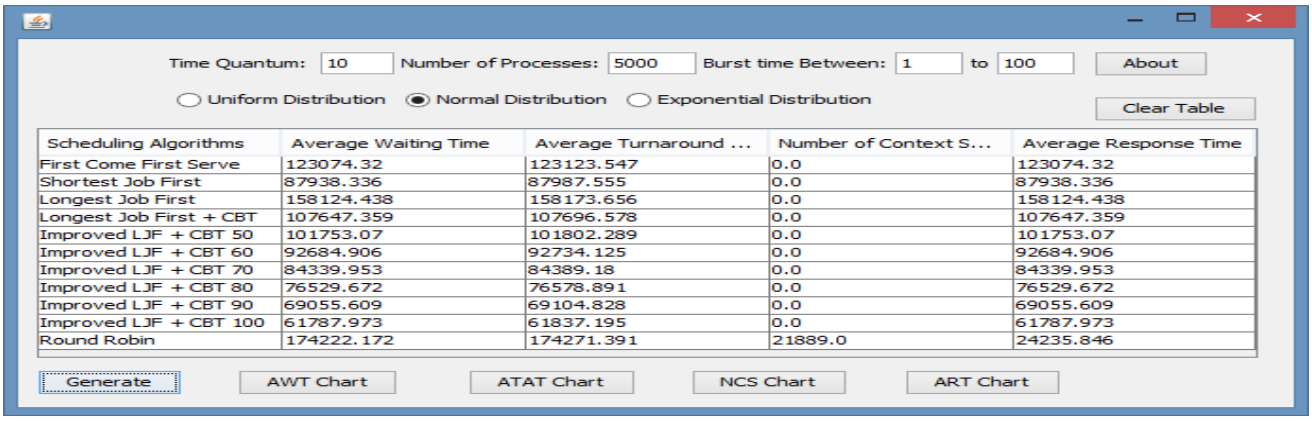


Figure 4. 4: Graph of ART for 5000 Processes Using Uniform Distribution

Figure 4.5 shows the system interface which presented some results computation using 5000 processes generated by the normal distribution with burst times ranging between 1ms and 100ms, and the time quantum of 10ms for RR



Scheduling Algorithms	Average Waiting Time	Average Turnaround ...	Number of Context S...	Average Response Time
First Come First Serve	123074.32	123123.547	0.0	123074.32
Shortest Job First	87938.336	87987.555	0.0	87938.336
Longest Job First	158124.438	158173.656	0.0	158124.438
Longest Job First + CBT	107647.359	107696.578	0.0	107647.359
Improved LJF + CBT 50	101753.07	101802.289	0.0	101753.07
Improved LJF + CBT 60	92684.906	92734.125	0.0	92684.906
Improved LJF + CBT 70	84339.953	84389.18	0.0	84339.953
Improved LJF + CBT 80	76529.672	76578.891	0.0	76529.672
Improved LJF + CBT 90	69055.609	69104.828	0.0	69055.609
Improved LJF + CBT 100	61787.973	61837.195	0.0	61787.973
Round Robin	174222.172	174271.391	21889.0	24235.846

Figure 4. 5: System Interface for 5000 Processes Using Normal Distribution

Figure 4.6 shows the graphical result of AWT of 5000 processes obtained in figure 4.5. It was observed that the proposed algorithm with 100thpercentile was better compared to others. This is because it produced lowest results i.e., 61787.973ms, followed by 90th percentile, then 80th percentile, 70th percentile, SJF, 60th percentile,50th percentile, LJF+CBT, FCFS, LJF and RR respectively.

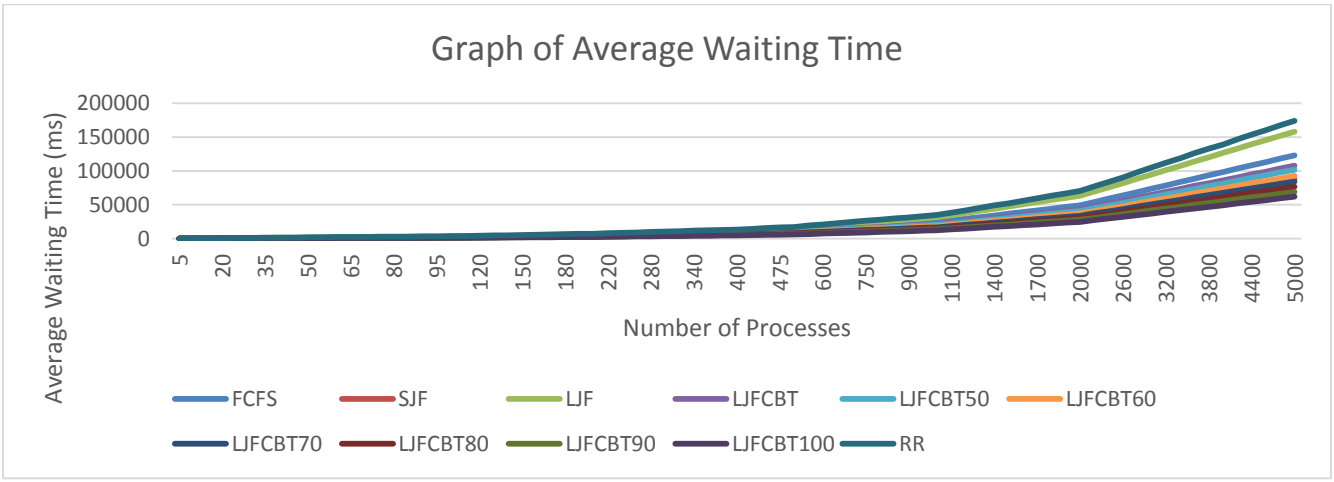


Figure 4. 6: Graph of AWT for 5000 Processes Using Normal Distribution

Figure 4.7 shows the graphical result of ATAT of 5000 processes obtained in Figure 4.5. It was observed that the proposed algorithm with 100thpercentile was better compared to others. This is because it produced lowest results i.e., 61837.195ms, followed by 90thpercentile, then 80th percentile, 70thpercentile, SJF, 60thpercentile, 50thpercentile, LJF+CBT, FCFS, LJF and RR respectively.

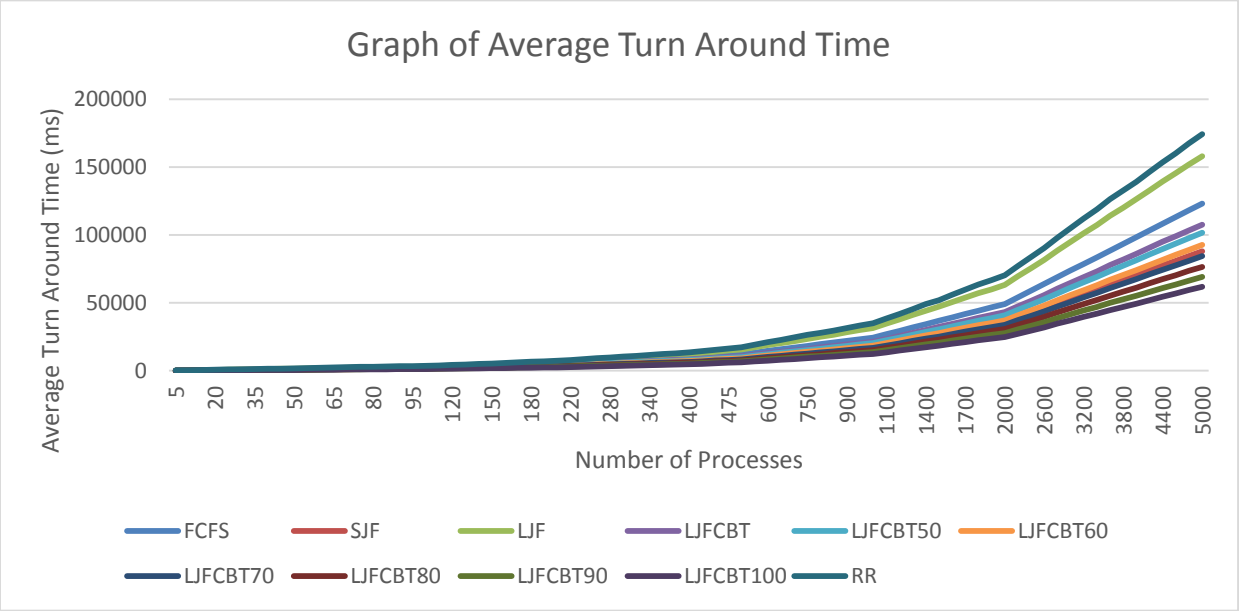


Figure 4. 7: Graph of ATAT for 5000 Processes Using Normal Distribution

Figure 4.8 shows the graphical result of ART of 5000 processes obtained in Figure 4.5. It was observed that the RR was better compared to others. This is because it produced lowest results i.e., 24235.846ms, followed by the proposed algorithm with 100thpercentile then 90th percentile,80th percentile, 70thpercentile, SJF, 60thpercentile, 50thpercentile, LJF+CBT, FCFS and LJF respectively.

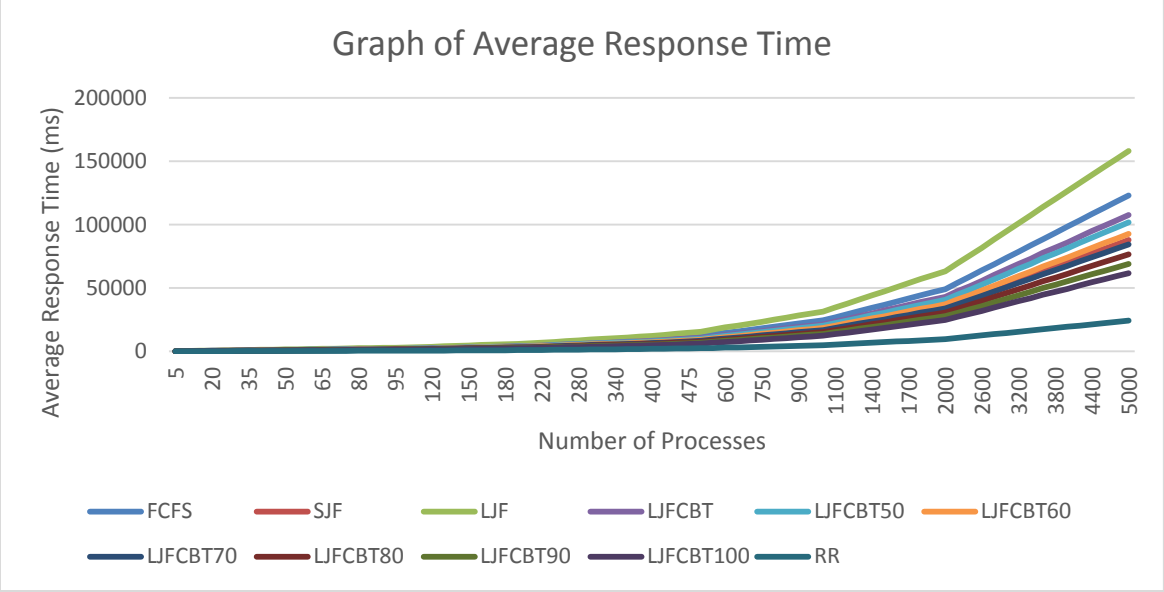
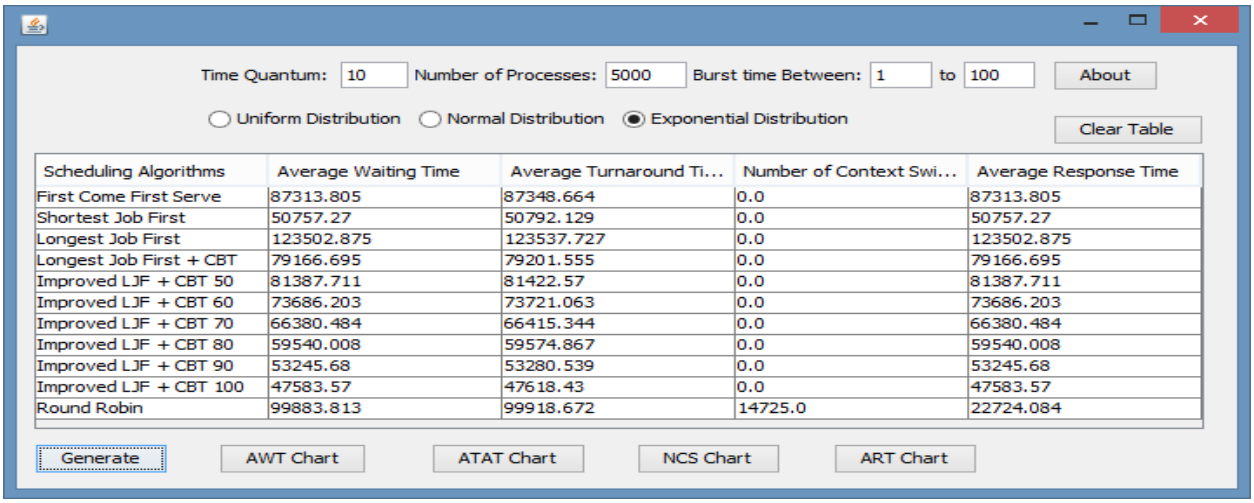


Figure 4. 8: Graph of ART for 5000 Processes Using Normal Distribution

Figure 4.9 shows the system interface which presented some results computation using 5000 processes generated by the exponential distribution with burst times ranging between 1ms and 100ms, and the time quantum of 10ms for RR



Scheduling Algorithms	Average Waiting Time	Average Turnaround Ti...	Number of Context Swi...	Average Response Time
First Come First Serve	87313.805	87348.664	0.0	87313.805
Shortest Job First	50757.27	50792.129	0.0	50757.27
Longest Job First	123502.875	123537.727	0.0	123502.875
Longest Job First + CBT	79166.695	79201.555	0.0	79166.695
Improved LJF + CBT 50	81387.711	81422.57	0.0	81387.711
Improved LJF + CBT 60	73686.203	73721.063	0.0	73686.203
Improved LJF + CBT 70	66380.484	66415.344	0.0	66380.484
Improved LJF + CBT 80	59540.008	59574.867	0.0	59540.008
Improved LJF + CBT 90	53245.68	53280.539	0.0	53245.68
Improved LJF + CBT 100	47583.57	47618.43	0.0	47583.57
Round Robin	99883.813	99918.672	14725.0	22724.084

Figure 4. 9: System Interface for 5000 Processes Using Exponential Distribution

Figure 4.10 shows the graphical result of AWT of 5000 processes obtained in Figure 4.9. It was observed that the proposed algorithm with 100thpercentile was better compared to others. This is because it produced lowest results i.e., 47583.57ms, followed by 90th percentile, then 80th percentile, 70th percentile, SJF, 60th percentile,50th percentile, LJF+CBT, FCFS, LJF and RR respectively.

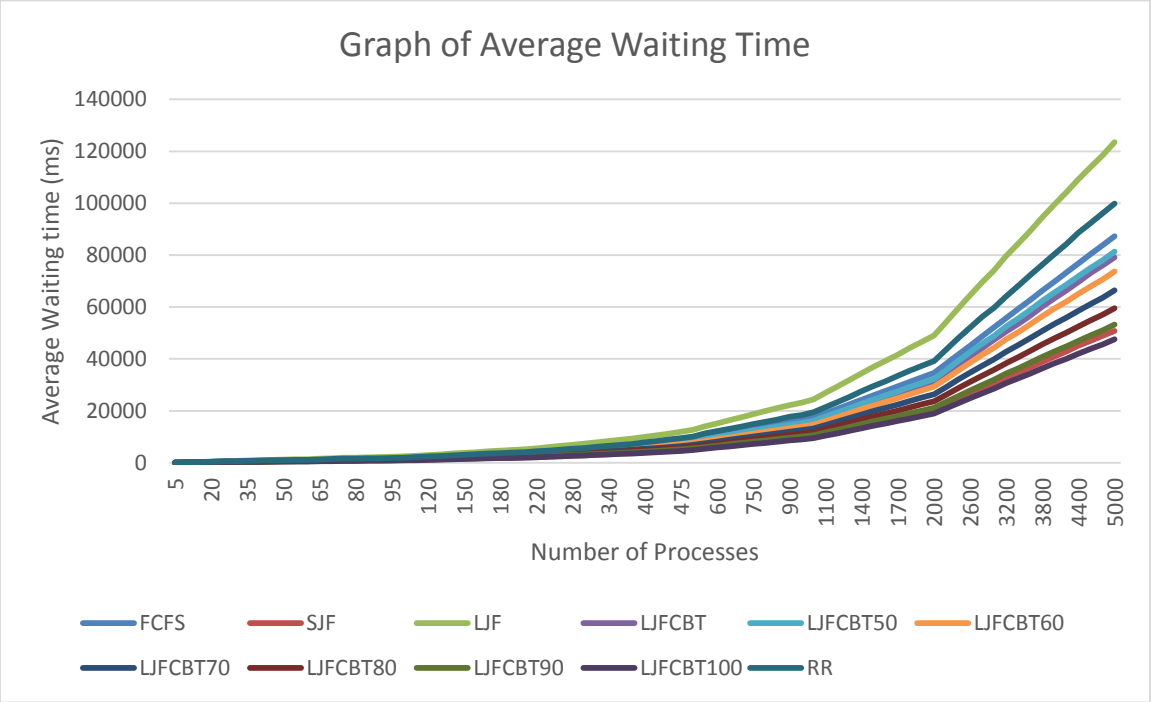


Figure 4. 10: Graph of AWT for 5000 Processes Using Exponential Distribution

Figure 4.11 shows the graphical result of ATAT of 5000 processes obtained in Figure 4.9. It was observed that the proposed algorithm with 100thpercentile was better compared to others. This is because it produced lowest results i.e., 47618.43ms, followed by 90th percentile, then 80th percentile, 70th percentile, SJF, 60th percentile,50th percentile, LJF+CBT, FCFS, LJF and RR respectively.

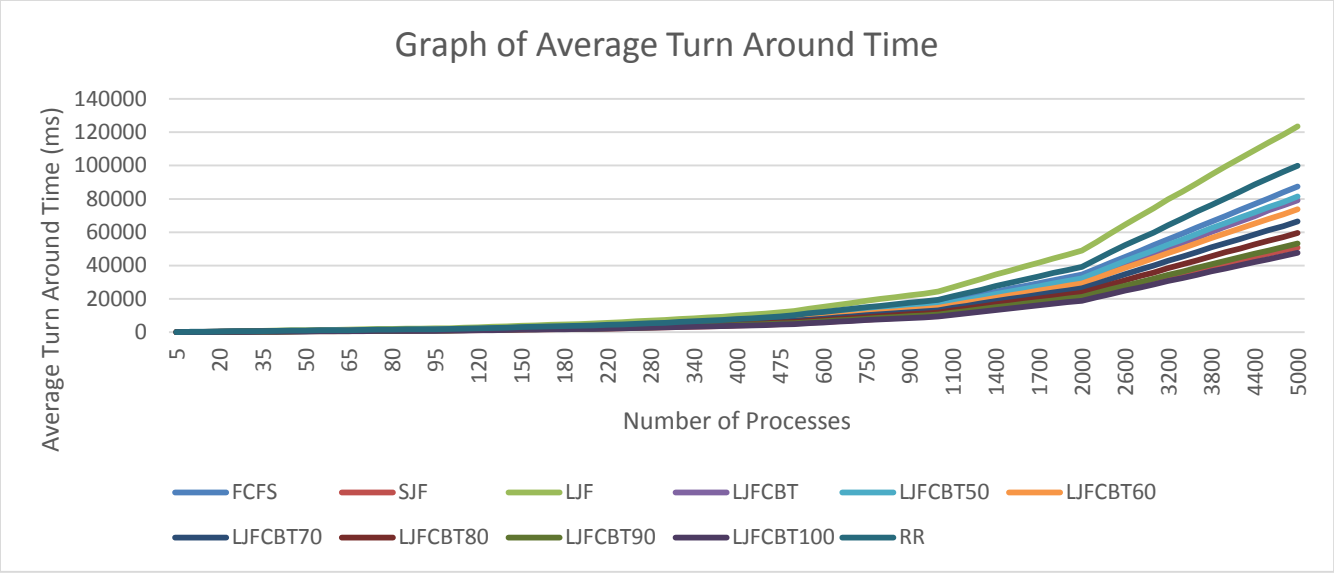


Figure 4. 11: Graph of ATAT for 5000 Processes Using Exponential Distribution

Figure 4.12 shows the graphical result of ART of 5000 processes obtained in Figure 4.9. It was observed that the RR was better compared to others. This is because it produced lowest results i.e., 22724.084ms, followed by proposed algorithm with 100thpercentile then 90th percentile,80th percentile, 70thpercentile, SJF, 60th percentile,50th percentile, LJF+CBT, FCFS and LJF respectively.

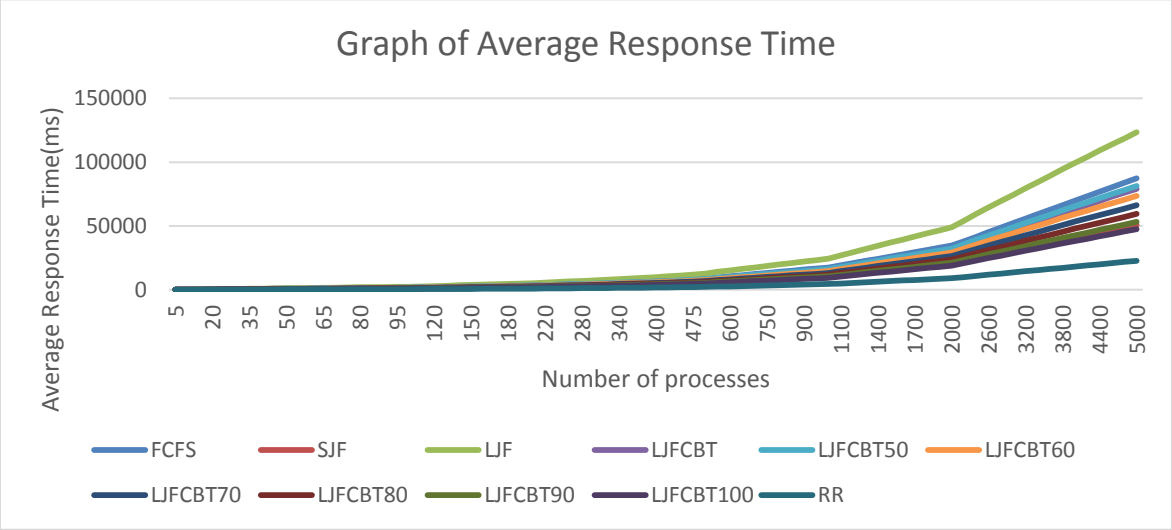


Figure 4. 12: Graph of ART for 5000 Processes Using Exponential Distribution

From the preceding graphs above, it is observed that with the uniform and normal distribution, the proposed 100th percentile algorithm produced the best Average Waiting Time (AWT) and Average Turnaround Time (ATAT), the proposed 90th percentile algorithm came second followed by proposed 80th, proposed 70th, SJF, proposed 60th, proposed 50th LJF+CBT, FCFS, LJF and RR respectively. RR produced the best Average Response Time (ART), the proposed 100th percentile algorithm came second, and then the proposed 90th percentile algorithm came second followed by proposed 80th, proposed 70th, SJF, proposed 60th, proposed 50th LJF+CBT, FCFS and LJF respectively.

With the exponential distribution, the proposed 100th percentile algorithm produced the best Average Waiting Time (AWT) and Average Turnaround Time (ATAT), SJF came second, followed by the proposed 90th percentile algorithm, proposed 80th, proposed 70th, proposed 60th, proposed 50th LJF+CBT, FCFS, LJF and RR respectively. RR produced the best Average Response Time (ART), the proposed 100th percentile algorithm came second, and then SJF, followed by the proposed 90th, proposed 80th, proposed 70th, proposed 60th, proposed 50th LJF+CBT, FCFS and LJF respectively.

From the results of the five scheduling algorithms (FCFS, SJF, RR, LJF and LJF+CBT) compared with the proposed algorithm (ILJF), it has shown that the proposed algorithm (ILJF) has better Average Waiting Time (AWT) and Average Turnaround Time (ATAT) against FCFS, SJF, RR, LJF and LJF+CBT. It also has better Average Response Time (ART) against FCFS, SJF, LJF+CBT and LJF.

A summary of the performance percentage difference between the compared algorithms is shown in Tables 4.1 for uniform distribution, 4.2 for normal distribution and 4.3 for exponential distribution based on AWT, ATAT and ART.

Table 4. 1: Performance Summary for Uniform Distribution

Algorithms	**Criteria**		
	AWT	**ATAT**	**ART**
FCFS	50%	50%	50%
SJF	26%	26%	26%
LJF	62%	62%	62%
LJF+CBT	44%	44%	44%
LJF+CBT50	41%	41%	41%
LJF+CBT60	35%	35%	35%
LJF+CBT70	27%	27%	27%
LJF+CBT80	19%	19%	19%
LJF+CBT90	10%	10%	10%
LJF+CBT100	0	0	0
RR	62%	62%	-163%

Table 4. 2: Performance Summary for Normal Distribution

Algorithms	**Criteria**		
	AWT	**ATAT**	**ART**
FCFS	50%	50%	50%
SJF	30%	30%	30%
LJF	61%	61%	61%
LJF+CBT	43%	43%	43%
LJF+CBT50	39%	39%	39%
LJF+CBT60	33%	33%	33%
LJF+CBT70	27%	27%	27%
LJF+CBT80	19%	19%	19%
LJF+CBT90	11%	11%	11%
LJF+CBT100	0	0	0
RR	65%	65%	-154%

Table 4. 3: Performance summary for Exponential Distribution

Algorithms	Criteria		
	AWT	**ATAT**	**ART**
FCFS	46%	45%	46%
SJF	6%	6%	6%
LJF	61%	61%	62%
LJF+CBT	40%	40%	40%
LJF+CBT50	42%	42%	42%
LJF+CBT60	35%	35%	35%
LJF+CBT70	28%	28%	28%
LJF+CBT80	20%	20%	20%
LJF+CBT90	11%	11%	11%
LJF+CBT100	0	0	0
RR	53%	53%	-109%

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

The aim of CPU scheduling is to assign a process to be executed by a processor over time, in a way that meets system objective, such as response time, throughput and processor efficiency. Longest Job First (LJF) CPU scheduling algorithm assigns the CPU to the process in queue with the highest CPU burst but it suffers from the problem of starvation of processes with short burst times. This research proposed an algorithm (LJF+CBT50, LJF+CBT60, LJF+CBT70, LJF+CBT80, LJF+CBT90, LJF+CBT100) based on the modification of LJF+CBT scheduling algorithm by (Abdullahi and Junaidu, 2013). Using the principle of percentile, a new method of determining the threshold was used for the categorization of processes and also a new method of merging the shorter processes in the short category was introduced. The proposed algorithm was implemented in Java. Using three statistical distributions (uniform distribution, normal distribution and exponential distribution), random burst times were generated for different set of processes which were used to compare the performance of the proposed algorithm with First Come First Serve, Shortest Job First, Round Robin, Longest Job First and Longest Job First with Combinational Burst Time Scheduling Algorithm based on AWT, ATAT, ART.

From the simulation results, it was observed that with the uniform and normal distribution, the proposed 100^{th} percentile algorithm produced the best Average Waiting Time (AWT) and Average Turnaround Time (ATAT), the proposed 90^{th} percentile algorithm came second followed by proposed 80^{th}, proposed 70^{th}, SJF, proposed 60^{th}, proposed 50^{th} LJF+CBT, FCFS, LJF and RR respectively. RR produced the best Average Response Time (ART), the proposed 100^{th} percentile algorithm came second, and then the proposed 90^{th} percentile algorithm came

second followed by proposed 80th, proposed 70th, SJF, proposed 60th, proposed 50th LJF+CBT, FCFS and LJF respectively.

With the exponential distribution, the proposed 100th percentile algorithm produced the best Average Waiting Time (AWT) and Average Turnaround Time (ATAT), SJF came second, followed by the proposed 90th percentile algorithm, proposed 80th, proposed 70th, proposed 60th, proposed 50th LJF+CBT, FCFS, LJF and RR respectively. RR produced the best Average Response Time (ART), the proposed 100th percentile algorithm came second, and then SJF, followed by the proposed 90th, proposed 80th, proposed 70th, proposed 60th, proposed 50th LJF+CBT, FCFS and LJF respectively.

From the results of the five scheduling algorithms (FCFS, SJF, RR, LJF and LJF+CBT) compared with the proposed algorithm (ILJF), it has shown that the proposed algorithm (ILJF) has better Average Waiting Time (AWT) and Average Turnaround Time (ATAT) against FCFS, SJF, RR, LJF and LJF+CBT. It also has better Average Response Time (ART) against FCFS, SJF, LJF+CBT and LJF.

5.2 CONCLUSION

Longest Job First(LJF) is considered among the most uncommon and inefficient scheduling algorithms, the major disadvantage of which is *starvation* of the short processes. This research dissertation proposed an enhanced algorithm, Improved LJF, based on the LJF+ CBT algorithm of (Abdullahi and Junaidu, 2013). From results of the experiments carried out, the proposed algorithm Improved LJF, improved on addressing the starvation problem thereby minimizing average waiting time, average turnaround time and Average Response Time.

5.3 RECOMMENDATION

This work assumed that the processes arrived at the same time. Therefore, the future research should investigate the behavior of the algorithm with random arrival time of processes.

REFERENCES

Abdullahi, I. and Junaid, S. B. (2013). Empirical Framework to Migrate Problems in Longer Job First Scheduling Algorithm (LJF+CBT). *International Journal of Computer Application, 75*(14): pp. 9-14.

Abdullahi, I. (2012). Process Scheduling In Longest Job First (LJF) Algorithm: *A Proposed Frame Work For Starvation Processes MSC Thesis.* Department of Mathematics: ABU,Zaria.

AbdulRazaq, A., Saleh, E. and Junaidu, S. (2014). A New Improved Round Robin (NIRR) CPU Scheduling Algorithm. *Internation Journal of Computer Application, 90*(4): pp. 27-33.

Abdulrazaq, A., Salisu, A. and Ahmad, M.M. (2014). An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(2): pp. 601-610.

Ali, D. J. (2012). Improving Efficiency of Round Robin Scheduling Using Ascending Quantum and Minimum-Maximum Burst Time. *Journal of University of Anbar for Pure Science, Ramadi*, 6(2): pp. 23-27.

Cheng, T. and Kahlbacher, H. (2006). A Proof of the Longest Job First Policy in one Machine scheduling. *CA:Naval Reasearch Logistics, 38*(5): pp.715-720.

Ernst, B. W., Schroeder, B and Urvoykella, G. (2007). Scheduling in Practice. *Journal of Parallel Processing, 34*(4), pp. 2-7.

Frederic, H. (2010). Retrieved May 2nd, 2015, from www.it.uu.se/edu/course/homepag/oskomp/vt07/lectures/scheduling-algorithms/handout.pdf: www.it.uu.se/edu/course/homepag/oskomp/vt07/lectures/scheduling-algorithms/handout.pdf

Ida, F. A and MacIver, A. M. (2011). *ftcc_operatingsystems/chapter_5a.html*. Retrieved May 2, 2015, from ftcc_operatingsystems/chapter_5a.html: http://www.oocities.org

Insup, L. (2007). Retrieved May 2, 2015, from www.cis.upenn.edu/~lee/07cis505/Lec/osschedule-: www.cis.upenn.edu/~lee/07cis505/Lec/osschedule-

Jerry, B., John, S. C., Barry, L. N and David, M. N. (2005). Discrete-Event System Simulation. (4th, Ed.) New Jersey, U.S.A: Pearson Education International, pp.239-244.

Normal Distribution. (n.d.). Retrieved October 10, 2015, from http://maths.berkeley.edu/~scanlon/m6b504/In/16b21ec31.pdf

Oladugba, A. V., Udom, A. U., Ugah, T. E., Ukaegbu, E. C., Madukaife, M. S., & Sanni, S. S. (2014). *Principles of Applied Statistics* (Revised ed.). Enugu, Nigeria: University of Nigeria Press Limited, pp. 239 -244.

Operating System Tutorial . (n.d.). Retrieved May 2nd, 2015, from www.tutorialspoint.com: http://www.tutorialspoint.com/operating_system/index.htm

Operating Systems. (n.d.). Retrieved may 2nd, 2015, from www.pling.org.uk: http://www.pling.org.uk/cs/ops.html

Operating Systems, *Lecture 5*. (n.d.). Retrieved February 28, 2016, from http://www.cs.kent.edu/~javed/class-OS10S/OS-AL05.pdf

Oyetunj, E. O and Oluyele, A. E. (2009). Performance Assessment of Some CPU Scheduling Algorithms. *Research Journal of Information Technology, 1*(1): pp.22-26.

Percentile and Percentile Rank, Chapter Five. (n.d.). Retrieved May 2nd, 2015, from harding.edu/sbreezeel/460%20files/statbook/chapter5.pdf: harding.edu/sbreezeel/460%20files/statbook/chapter5.pdf

Rahul, S and Umesh, C. J. (2012). Process Scheduling Using Repeatative Subtraction (An Algorithm dependent on the number of processes executed since its Arrival. *International Journal of Advanced Research in Computer Science and Software Engineering, 2*(11): pp. 425-429.

Rakesh, M., Manas, D., Lakshmi, P. M and Sudhashree, A. (2011). Design and Performance Evaluation of A New Proposed Fittest Job First Dynamic Round Robin (FJFDRR) Scheduling Algorithm. *International Journal of Computer Information Systems, 2*(2): pp. 23-27.

Rumsey, J. D. (2015). Statistic For Dummies (2nd ed.). New Jersey, U.S.A: John Wiley & Sons, Inc, pp. 19-22.

Silberschatz, A., Galvin, P. B and Gagne, G. (2005). Operating System Concepts (7th ed.). New Jersey, U.S.A: John Wiley and Sons Inc. pp. 1-885.

Simon, A., Saleh, S.E. and Junaidu, S.B. (2014). Dynamic Round Robin with Controlled Preemption (DRRCP). *International Journal of Computer Science Issues, 11*(3), pp. 109-117.

Simon, A., Saleh, S.E. and Junaidu, S.B. (2014). Half Life Variable Quantum Time Round Robin(HLVQTRR) CPU Scheduling Algorithm (IJCSIT). *International Journal of Computer Science and Information Technologies, 5*(6): pp. 7210-7217.

Soraj H and Roy K,C. (2012). Adaptive Round Robin Scheduling Using Shortest Burst Approach Based on Smart Time Slice.Retrived December 10,2012 *International Journal of Data Engineering*: 21(1): pp. 217-228.

Stallings, W. (2012). Operating System Internals and Design Principles (7th ed.). New Jersey, U.S.A: Prentice Hall, pp. 396-425.

Suri, K. P and Sumit, M. (2012). Design of Stochastic Simulator for Analyzing the Impact of Scalability on CPU Scheduling Algorithms. *International Journal of ComputerApplications, 49*(17): pp. 4-9.

Tanenbaum, A. S. (2009). Mordern Operating System (3rd ed.). New Jersey, U.S.A: pearson Education,Inc pp. 1104.

The Exponential Distribution. (n.d.). Retrieved October 10, 2015, from https://reliawiki.org/index.php/the_exponential_distribution

Uniform Distribution. (n.d.). Retrieved October 10, 2015, from https://www.itl.nist.gov/div898/handbook/eda/section3/eda3662.htm

Vissarion, F. (2011). Study of the Effect of Cost Policies in the convergence of selfish strategies in Pure Nash Equilibrum in congestion games. Retrieved October 2011, from www.users.uoa.gr: http://users.uoa.gr/~vfisikop/projects/thesis.pdf