DEVELOPMENT OF AN OPTIMIZED PRIORITY BASED CPU SCHEDULING ALGORITHM TO MINIMIZE STARVATION OF PROCESSES USING AN EFFICIENCY FACTOR

BY

MUHAMMAD AHMAD MUSTAPHA

DEPARTMENT OF MATHEMATICS

AHMADU BELLO UNIVERSITY,

ZARIA, NIGERIA

FEBRUARY, 2016.

DEVELOPMENT OF AN OPTIMIZED PRIORITY BASED CPU SCHEDULING ALGORITHM TO MINIMIZE STARVATION OF PROCESSES USING AN EFFICIENCY FACTOR

BY

Muhammad Ahmad Mustapha B.Tech Computer Science (ATBU) 2005

MSc/SCIE/08820/2009-2010

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

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

DEPARTMENT OF MATHEMATICS,

FACULTY OF SCIENCE

AHMADU BELLO UNIVERSITY,

ZARIA, NIGERIA

FEBRUARY, 2016

DECLARATION

I declare that the work in this project report entitled DEVELOPMENT OF AN OPTIMIZED PRIORITY BASED CPU SCHEDULING ALGORITHM TO MINIMIZE STARVATION OF PROCESSES USING AN EFFICIENCY FACTOR has been performed by me in the Department of Mathematics under the supervision of Dr. S.E Abdullahi and Prof. S. B Junaidu. The information derived from the literature has been duly acknowledged in the text and list of references provided. No part of this project report was previously presented for another degree or diploma at any university.

MUHAMMAD AHMAD MUSTAPHA		
Name of student	Signature	Date

CERTIFICATION

This dissertation entitled DEVELOPMENT OF AN OPTIMIZED PRIORITY BASED CPU SCHEDULING ALGORITHM TO MINIMIZE STARVATION OF PROCESSES USING AN EFFICIENCY FACTOR by MUHAMMAD AHMAD MUSTAPHA 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) _____

DR S. E. ABDULLAHI

Date _____

_____

Chairman, Supervisory Committee

(Signature) _____

PROF. S. B. JUNAIDU

Date _____

_____

Member, Supervisory Committee

(Signature) _____

Date _____

_____

Member Supervisory Committee

(Signature) _____

Date _____

_____

Head of Department

(Signature) _____

Date _____

_____

Dean, School of Postgraduate Studies

DEDICATION

This dissertation is dedicated to The Holy Prophet Muhammad (SAW) who holds the pen of the scholar in a higher esteem than the blood of the martyr and to whom the first word revealed was read.

ACKNOWLEDGEMENT

First and foremost my eternal gratitude goes to the Gracious Creator, Who gave me life and the opportunity to undertake this study and write this dissertation.

Secondly, I acknowledge my esteemed supervisors in the persons of Dr. S.E. Abdullahi and Prof. S. B. Junaidu who guided me throughout the period of writing this dissertation and without whose help and directions this dissertation will not have been written.

Then my parents and relatives who gave me all the support needed in the course of writing this dissertation, I pray that Almighty Allah reward their efforts.

Also not to forget my classmates and colleagues who helped me with useful advice in the course of writing this dissertation like Malam Buhari Abdullahi Sulaiman, Malam Salisu Aliyu, Malam Abdulrazaq Abdulrahim, Malam Aliyu Kufena and Malam Dishing among others in the Department of Mathematics Ahmadu Bello University Zaria.

Lastly, to my wife Malama Hauwa Abubakar Yusuf for being patient with me throughout the period she missed me in course of writing this dissertation, thank you very much, God bless.

ABSTRACT

The priority based CPU scheduling algorithm (i.e. Shortest Job First (SJF) or Priority Scheduling (PS)) is a kind of scheduling algorithm that assigns the CPU to processes based on the priority of each process. The shortcoming of both of these algorithms is starvation (i.e. starvation of processes with longer burst times in the case of SJF and starvation of processes with lower priorities in the case of PS). This dissertation proposes a new algorithm that introduces the concept of EFFICIENCY FACTOR which uses three properties which are priority, burst time and arrival time to compute one factor which is used to schedule processes in the case of the proposed Priority algorithm instead of only one factor that priority is in the case of traditional Priority algorithm and two properties which are burst time and arrival time to compute one factor which is used to schedule processes in the case of proposed Shortest Job First algorithm instead only one factor that is burst time in the case of traditional Shortest Job First algorithm. This proposed algorithm was implemented and benchmarked against SJF, PS and the Optimum Service Time Concept for Round Robin Algorithm (OSTRR) by Saxena and Agarwal (2012) using three different statistical distributions (namely Normal, Uniform and Exponential Distributions) to generate the burst times of the processes, two different statistical distributions (namely Uniform and Exponential Distributions) to generate the priorities of the processes and also, uses Poisson Distribution to generate the arrival times of processes. It is observed that in the SJF category, the traditional SJF produced better Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared with the proposed SJF. But they both produced the same Number of Context Switches (NCS). The proposed SJF produced better results compared with OSTRR with respect to AWT, ATAT, ART, NCS and WTVD. While in the PS category, the proposed Priority produced better AWT, ATAT, ART and WTVD compared to the traditional Priority

scheduling algorithm. But they both produced the same NCS. The proposed Priority algorithm produced better results which means the proposed priority algorithm produced lesser NCS and smaller value of WTVD when compared to OSTRR by Saxena and Agarwal (2012) and OSTRR by Saxena and Agarwal (2012) produced minimal AWT and ATAT than the proposed Priority algorithm although the values are almost identical in all categories of the statistical distributions used. Based on these results, the proposed Priority algorithm should be preferred over the traditional Priority algorithm. This is because, starvation has been minimized by reducing the average waiting time, average turnaround time, average response time and waiting time variance deviation in the case of the 500 processes that were considered and the burst time of processes ranging between 1 and 100ms and a quantum time of 10ms used by OSTRR by Saxena and Agarwal (2012).

TABLE OF CONTENTS

TITLE PAGE……i
DECLARATION ii
CERTIFICATION iii
DEDICATION iv
ACKNOWLEDGEMENT v
ABSTRACT vi
TABLE OF CONTENTS viii
LIST OF TABLES x
LIST OF FIGURES xi
LIST OF ABBREVIATIONS xiii
CHAPTER ONE: INTRODUCTION 1
1.1 Background of the Study 1
1.2 Research Motivation and Goals 2
1.3 Research Aim and Objectives 2
1.4 Research Methodology 3
1.5 Contribution to knowledge 5
CHAPTER TWO: LITERATURE REVIEW 6
2.1 INTRODUCTION 6
2.2 OPERATING SYSTEM 6
2.1.1 The Process State 7
2.1.2 CPU Scheduling Algorithms 8
2.1.3 Performance Criteria 10
2.3 Waiting Time Variance 11
2.4 CPU Scheduling 12
2.4.1 The importance of priority in CPU Scheduling 13
2.5 Review of Priority Based CPU Scheduling Algorithms 13
2.6 Observations on the Review Literatures Showing the Knowledge Gap Addressed by the Dissertation 18
2.7 Waiting Time Variance (WTV): 19
CHAPTER THREE: MATERIALS AND METHOD 21
3.1 Introduction 21
3.2 Assumption 21

3.3 Efficiency Factor 21
3.4 The pseudo code of the Proposed Priority Based CPU Scheduling Algorithms 23
3.5 Proposed Algorithm for Priority CPU Scheduling Algorithm 27
3.6 Proposed Algorithm for Shortest Job First CPU Scheduling Algorithm 30
3.7 Time Complexity of the Proposed Priority Based CPU Scheduling Algorithm 33
CHAPTER FOUR: RESULTS AND DISCUSSION 35
4.1 Introduction 35
4.2 Illustrative Examples 35
4.2.1 Shortest Job First (SJF) 36
4.2.2 Priority Scheduling (PS) 38
4.2.3 Proposed Shortest Job First 39
4.2.4 Proposed Priority Scheduling 40
4.2.5 OSTRR 41
4.3 System Requirements 43
4.3.1 Experimental Setup 43
4.4 Results and Discussion 45
4.4.1 Results obtained using normal distribution to generate burst time 46
4.4.1.1 First category 47
4.4.1.2 Second category 53
4.4.2 Results obtained using uniform distribution to generate burst time 58
4.4.2.1 First category 59
4.4.2.2 Second category 65
4.4.3 Results obtained using exponential distribution to generate burst time 70
4.4.3.1 First category 71
4.4.3.2 Second category 77
CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION 85
5.1 Summary 85
5.2 Conclusion 85
5.3 Recommendation 88
REFERENCES 89
APPENDIX 92

LIST OF TABLES

Table 3.1: Process Table 28
Table 3.2: Process Table for Priority CPU Scheduling Algorithm 30
Table 3.3: Process Table 31
Table 3.4: Process Table for Shortest Job First CPU Scheduling Algorithm 33
Table 4.1: Process Table 35
Table 4.2: Comparative Table 43
Table 4.3: Performance of Statistical Distributions in Generating Burst Times 82
Table 4.4: Performance of Algorithms Based on Metrics Used in Comparing Them for First Category 83
Table 4.5: Performance of Algorithms Based on Metrics Used in Comparing Them for Second Category 84

LIST OF FIGURES

Figure1.1: The Traditional process state diagram..... 2
Figure2.1: Memory Layouts in Multiprocessing System 7
Figure4. 1: Gantt chart representation of SJF 36
Figure4. 2: Gantt chart representation of Priority CPU Scheduling Algorithm 38
Figure4. 3: Gantt chart representation of Proposed Shortest Job First 39
Figure4. 4: Gantt chart representation of Proposed Priority Scheduling 40
Figure4. 5: Gantt chart representation of OSTRR 41
Figure 4. 6: System Interface with some Process Execution in Progress 46
Figure4. 7: Data for Average Waiting Time and Average Turnaround Time using Normal Distribution for Shortest Job First..... 47
Figure4. 8: Graph of Average Waiting Time using Normal Distribution..... 47
Figure4. 9: Graph of Average Turnaround Time using Normal Distribution..... 48
Figure4. 10: Data for Average Response Time and Number of Context Switches using Normal Distribution for Shortest Job First..... 49
Figure4. 11: Graph of Average Response Time using Normal Distribution 50
Figure4. 12: Graph of Number of Context Switches using Normal Distribution 50
Figure4. 13: Data for Waiting Time Variance Deviation using Normal Distribution for Shortest Job First..... 51
Figure4. 14: Graph of Waiting Time Variance Deviation 51
Figure4. 15: Data for Average Waiting Time and Average Turnaround Time using Normal Distribution for Priority 53
Figure4. 16: Graph of Average Waiting Time using Normal Distribution..... 53
Figure4. 17: Graph of Average Turnaround Time using Normal Distribution..... 54
Figure4. 18: Data for Average Response Time and Number of Context Switches using Normal Distribution for Priority 55
Figure4. 19: Graph of Average Response Time using Normal Distribution 56
Figure4. 20: Graph of Number of Context Switches using Normal Distribution 56
Figure4. 21: Data for Waiting Time Variance Deviation using Normal Distribution for Priority 57
Figure4. 22: Graph of Waiting Time Variance Deviation 58
Figure4. 23: Data for Average Waiting Time and Average Turnaround Time using Uniform Distribution for Shortest Job First..... 59
Figure4. 24: Graph of Average Waiting Time using Uniform Distribution 59
Figure4. 25: Graph of Average Turnaround Time using Uniform Distribution 60
Figure4. 26: Data for Average Response Time and Number of Context Switches using Uniform Distribution for Shortest Job First..... 61
Figure4. 27: Graph of Average Response Time using Uniform Distribution..... 62
Figure4. 28: Graph of Number of Context Switches using Uniform Distribution 62
Figure4. 29: Data for Waiting Time Variance Deviation using Uniform Distribution for Shortest Job First..... 63
Figure4. 30: Graph of Waiting Time Variance Deviation using Uniform Distribution 63
Figure4. 31: Data for Average Waiting Time and Average Turnaround Time using Uniform Distribution for Priority 65
Figure4. 32: Graph of Average Waiting Time using Uniform Distribution 65
Figure4. 33: Graph of Average Turnaround Time using Uniform Distribution 66

Figure4. 34: Data for Average Response Time and Number of Context Switches using Uniform Distribution for Priority 67
Figure4. 35: Graph of Average Response Time using Uniform Distribution..... 68
Figure4. 36: Graph of Number of Context Switches using Uniform Distribution 68
Figure4. 37: Data for Waiting Time Variance Deviation using Uniform Distribution for Priority 69
Figure4. 38: Graph of Waiting Time Variance Deviation using Uniform Distribution 69
Figure4. 39: Data for Average Waiting Time and Average Turnaround Time using Exponential Distribution for Shortest Job First..... 71
Figure4. 40: Graph of Average Waiting Time using Exponential Distribution 71
Figure4. 41: Graph of Average Turnaround Time using Exponential Distribution 72
Figure4. 42: Data for Average Response Time and Number of Context Switches using Exponential Distribution for Priority..... 73
Figure4. 43: Graph of Average Response Time using Exponential Distribution..... 73
Figure4. 44: Graph of Number of Context Switches using Exponential Distribution..... 74
Figure4. 45: Data for Waiting Time Variance Deviation using Exponential Distribution for Shortest Job First..... 75
Figure4. 46: Graph of Waiting Time Variance Deviation using Exponential Distribution..... 75
Figure4. 47: Data for Average Waiting Time and Average Turnaround Time using Exponential Distribution for Priority 77
Figure4. 48: Graph of Average Waiting Time using Exponential Distribution 77
Figure4. 49: Graph of Average Turnaround Time using Exponential Distribution 78
Figure4. 50: Data for Average Response Time and Number of Context Switches using Exponential Distribution for Priority..... 79
Figure4. 51: Graph of Average Response Time using Exponential Distribution..... 79
Figure4. 52: Graph of Number of Context Switches using Exponential Distribution..... 80
Figure4. 53: Data for Waiting Time Variance Deviation using Exponential Distribution for Priority 80
Figure4. 54: Graph of Waiting Time Variance Deviation using Exponential Distribution..... 81

LIST OF ABBREVIATIONS

ART – Average Response Time

AT – Arrival Time

ATAT – Average Turnaround Time

AWT – Average Waiting Time

BS – Balanced Spiral

BT – Burst Time

BW – Burst Time Priority Weight

BTW – Burst Time Weight

CPU – Central Processing Unit

DQRRR – Dynamic Quantum with Re-adjusted Round Robin

E – Efficiency Factor

E&C 1.1 – Eilon and Chowdhury 1.1

E&C 1.2 – Eilon and Chowdhury 1.2

f – Fit Factor

FCFS – First Come First Served

FIFO – First In First Out

HRRN – Highest Response Ratio Next

I/O –Input – Output

NCS – Number of Context Switches

OP – Optimum Priority

OST – Optimum Service Time

OSTRR – Optimum Service Time for Round Robin

PBSRR – Priority Based Static Round Robin

PCB – Process Control Block

PRIO - Priority

PS – Priority Scheduling

PW – Priority Weight

RR – Round Robin

SJF – Shortest Job First

SJN – Shortest Job Next

SP – Shortest Burst Time Priority

SPT – Shortest Processing Time

SRR – Selfish Round Robin

STS – Smart Time Slice

UP – User Priority

UW – User Priority Weight

VS – Verified Spiral

WTV – Waiting Time Variance

WTVD – Waiting Time Variance Deviation

WW – Waiting Time Weight

CHAPTER ONE: INTRODUCTION

This chapter discusses the introductory part of this dissertation which comprises the background of the study, research motivation and goals which stipulate the research questions for which the dissertation provides answers, the aims and objectives of the research work, the methodology that is followed in the research and the contribution which the dissertation makes to knowledge.

1.1 Background of the Study

The Central Processing Unit (CPU) is an important component of the computer system; hence it must be utilized efficiently. This can be achieved through what is called CPU scheduling (Oyetunji and Oluleye, 2009). CPU Scheduling refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. The CPU scheduling is one of the most important tasks of the operating system (Mehdi *et al*, 2012). The need for a scheduling algorithm to achieve the efficiency of the CPU arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously). The basic idea is to keep the CPU busy as much as possible by executing a user process or job until it must wait for an event, and then switch to another process. In multiprogramming systems, when there is more than one runnable process (that is ready to run on the CPU), the operating system must decide which one to activate as shown in Figure 1.1. The decision is made by the part of the operating system called the scheduler, using a scheduling algorithm (Suri and Sumit, 2012).

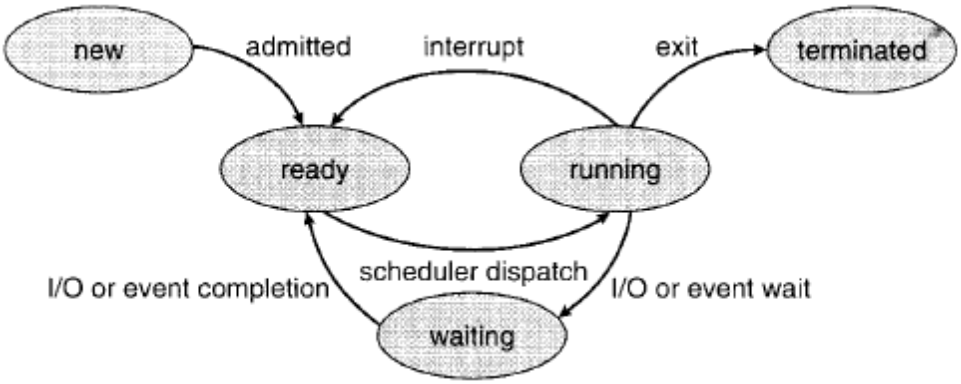


Figure1.1: The Traditional Process State Diagram (Silberschatz,*et al.* 2006)

1.2 Research Motivation and Goals

Priority based CPU Scheduling Algorithms such as Shortest Job First Algorithm which assigns CPU to processes according to their burst times and Priority CPU Scheduling Algorithm which assigns CPU to processes according to their priorities suffer from the problem of starvation. They are not fair as they are biased to processes of high priorities (Saxena and Agarwal 2012).

Consequently the goal of this work is to minimize starvation in the priority based CPU scheduling algorithms that is Shortest Job First and Priority scheduling algorithms thereby reducing the average waiting time, average turnaround time, average response time, number of context switches and waiting time variance deviation.

1.3 Research Aim and Objectives

The aim of this dissertation is to develop an algorithm that minimizes starvation in priority based CPU scheduling algorithms. This has been done by modifying the optimum priority in Saxena and Agarwal (2012) and applying it to the priority based CPU scheduling algorithms in order to

reduce average waiting time, average turnaround time, average response time and waiting time variance deviation. This is achieved with the following objectives:

1. Develop a new algorithm based on OSTRR by Saxena and Agarwal (2012)
2. Implement the algorithm using Java programming language
3. Assess the algorithm side-by-side traditional Shortest Job First algorithm, traditional Priority algorithm and OSTRR by Saxena and Agarwal (2012)

1.4 Research Methodology

The research was conducted in two stages as follows:

In the first stage, the following steps were taken

i. Values were assigned for process number, burst time, arrival time and priority randomly.
ii. The values assigned in i were used to calculate EFFICIENCY FACTORS for the processes both in the SJF and Priority categories.
iii. The processes were scheduled based on their EFFICIENCY FACTORS values.
iv. Compared the results for the proposed algorithm against the traditional priority based CPU scheduling algorithms and OSTRR by Saxena and Agarwal (2012) by the use of Gantt charts

In the second stage,

i. Developed an algorithm to reduce starvation in priority based CPU scheduling algorithms
ii. The data to be used was generated randomly using three different experiments as follows:
    a. In the first experiment, the arrival time is generated using Poisson Distribution (because it is a probability distribution of a discrete random variable that stands

for the number (count) of statistically independent events, occurring within a unit of time or space Letkowski (2012)); while the burst time is generated using Normal/Gaussian Distribution (because it is the most widely known and used of all distributions. And also, it approximates many natural phenomena so well, it has developed into a standard of reference for many probability problems (https://math.berkeley.edu/~scanlon/m16bs04/ln/16b2lec31.pdf)) and the priority is generated using Exponential Distribution (because it is used to model the behavior of units that have a constant failure rate (http://reliawiki.org/index.php/The_Exponential_Distribution)).

b. In the second experiment, the arrival time is generated using Poisson Distribution, the burst time is generated using Uniform Distribution (because it defines equal probability over a given range for a continuous distribution. For this reason, it is important as a reference distribution. It is 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 (http://www.itl.nist.gov/div898/handbook/eda/section3/eda3662.htm)) and the priority is generated using Exponential Distribution

iii. In the third experiment, the arrival time is generated using Poisson Distribution, the burst time is generated using Exponential Distribution and the priority is generated using Uniform Distribution.

Iii Simulated the algorithm developed in step i against the traditional priority based CPU scheduling algorithms and Optimum Service Time for Round Robin Algorithm(OSTRR) by Saxena and Agarwal (2012) using Java programming language,

iv. The comparison is done based on average waiting time, average turnaround time, average response time, number of context switches and waiting time variance deviation for the three different experiments on the priority based CPU scheduling algorithms and OSTRR by Saxena and Agarwal (2012)

1.5 Contribution to knowledge

The dissertation contributed to knowledge by optimizing the traditional priority CPU scheduling algorithms thereby reducing starvation using Efficiency Factor which makes use of three metrics that is arrival time, waiting time and priority of processes to obtain one optimal value to schedule each process instead of using only one metric that is the priority of a process to schedule the process in the case of traditional Priority algorithm.

CHAPTER TWO: LITERATURE REVIEW

2.1 INTRODUCTION

This chapter discusses the importance of the operating system as regards CPU scheduling, process state, CPU scheduling algorithms, performance criteria and waiting time variance. It also reviews some of the most important researches in the literature pertaining to the priority based scheduling algorithms, and use of waiting time variance in analyzing scheduling algorithms.

2.2 OPERATING SYSTEM

An introduction to operating system as an area of study will lay better foundations to our research as Galvin *et al*, (2005) defined Operating system as a program that manages the computer hardware. Operating system provides foundations and serves as intermediary between the computer system hardware and the user. (Stallings, 2001)

Operating system can also be looked at to perform three (3) principal functions which are (Abdullahi, 2012):-

1. Allocating and assigning of system resources such as I/O devices, software, Central Processing Units (CPU), memory etc.
2. Scheduling: - The function that coordinates resources and jobs using some selected algorithms.
3. Monitoring: - The function that monitors and keep tracks of activities in the computer system.

Generally, one of the major goals of the operating system is the maximization of the CPU utility and to achieve this, the concept of multiprogramming was conceived (Abdullahi, 2012) as illustrated in Figure 2.1

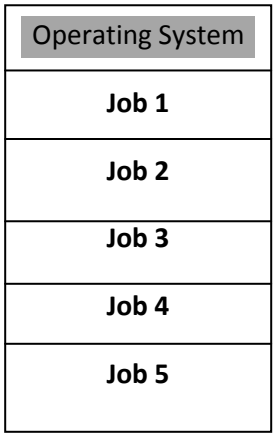


Figure2.1: Memory Layouts in Multiprocessing System

2.2.1 The Process State

There are three possible process states: running state (process is running on the CPU), ready state (process is ready to run but not actually running on the CPU) and waiting state (process is waiting for some event like I/O to happen). The CPU chooses (schedules) which process to run when any of the following occurs (Silberschatz ,*et al.* 2006):

1. When process switches from running to waiting. This could be as a result of I/O request, waiting for child to terminate, or waiting for synchronization operation to complete.
2. When process switches from running to ready. This could be as a result of completion of interrupt handler. If scheduler switches processes in this case, it has preempted the running process.
3. When process switches from waiting to ready state (on completion of I/O).
4. When a process terminates.

A number of basic assumptions are often being made in CPU scheduling. These are as follows:

1. There is a pool of runnable processes contending for the CPU.
2. The processes are independent and compete for resources.

The job of the scheduler is to distribute the scarce resources of the CPU to the different processes fairly (according to some definition of fairness) and in a way that optimizes some performance criteria (Silberschatz, *et al.* 2006).

2.2.2 CPU Scheduling Algorithms

CPU Scheduling is the act of selecting the next process for the CPU to service, once the current process leaves the CPU idle. Some basic CPU scheduling algorithms are listed below:

1. First-Come First-Serve (FCFS)

By far the simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this algorithm, processes are assigned the CPU in the order they request it. Basically, there is a single queue of ready processes. Relative importance of jobs measured only by arrival time (poor choice). The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. The average waiting time under the FCFS policy, however, is often quite long (http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node122.html).A long CPU-bound job may dominate the CPU and may force shorter jobs to wait for prolonged periods.

2. Shortest-Job-First (SJF)

This is a priority based algorithm which associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used. The SJF scheduling algorithm gives the minimum average waiting time for a given set of processes. Moving a short process before a long one, decreases the waiting time of the short process more than it increases the waiting time of the long process. Consequently, the average

waiting time decreases. The real difficulty with the SJF algorithm is knowing the length of the next CPU request. For long-term (job) scheduling in a batch system, we can use as the length the process time limit that a user specifies when he submits the job. Although the SJF algorithm is optimal, it cannot be implemented at the level of short-term CPU scheduling. There is no way to know the length of the next CPU burst. We may not know the length of the next CPU burst, but we may be able to predict its value. We expect that the next CPU burst will be similar in length to the previous ones. Also, long running jobs may starve for the CPU when there is a steady supply of short jobs (http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node123.html). The SJF is a priority based scheduling algorithm which assigns higher priority to shorter jobs (https://www.it.uu.se/edu/course/.../scheduling_algorithms/handout.pdf*)*.

$$priority_i = \frac{1}{burst\ time_i} \qquad (2.1)$$

i= $1\ to\ n, where\ n\ is\ the\ number\ of\ processes$, I is the i^{th} process

3. Priority Scheduling (PS)

A priority number is associated with each process. The CPU is allocated to the process with the highest priority. If there are multiple processes with same priority, then FCFS is used to allocate the CPU. A variation of this scheme allows preemption of the current process whenever a higher priority process arrives. Another variation of the policy adds an aging scheme, where the priority of a process increases as it remains in the ready queue (Silberschatz,*et al.* 2006).

4. Round-Robin Scheduling (RR)

This algorithm is especially designed for time-sharing systems; each process gets a small unit of CPU time. This algorithm allows the first process in the queue to run until it expires its time, and then runs the next process in the queue. In a situation where the process needs more time, the process runs for the full length of the time quantum and then it is preempted and then added to the tail of the queue (Silberschatz,*et al.* 2006).

2.2.3 Performance Criteria

The various CPU scheduling algorithms 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*, 2006):

1. 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, leads to wastage of time and memory, which in turn increases the overhead of the scheduler, so the design of operating system is to optimize only these switches.
2. Throughput: Throughput is defined as number of processes completed per unit time. Throughput is low in round robin scheduling implementation. Context switching and throughput are inversely proportional to each other.
3. CPU Utilization: This is a measure of how much busy the CPU is. Usually, the goal is to maximize the CPU utilization.

4. Turnaround Time: Turnaround time refers to the total time which is spent to complete the process and is how long it takes the CPU to execute that process. The time interval from the time of submission of a process to the time of completion is the turnaround time.
5. Waiting Time: Waiting time is the total time a process has been waiting in ready queue.
6. 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 it can be concluded that a good scheduling algorithm for real time and time sharing systems must possess the following characteristics (Silberschatz,*et al.* 2006):

1. Minimum context switches.
2. Maximum CPU utilization.
3. Maximum throughput.
4. Minimum turnaround time.
5. Minimum waiting time.
6. Minimum response time.

2.3 Waiting Time Variance

Variance is a statistical word, which is the measurement of the spread between numbers in a data set. It measures how far each number in the set is from the mean. It is calculated by taking the differences between each number in the set and the mean, squaring the differences (to make them positive) and dividing the sum of the squares by the number of values in the set.

$$\sigma^2 = \frac{\sum(i-\mu)^2}{N} \quad (2.2)$$

$$where\ \sigma^2\ is\ the\ variance, N\ is\ the\ number\ of\ processes, i\ the\ i^{th}\ process$$

$$and\ \mu\ is\ the\ mean(\ that\ is\ the\ average)\ of\ the\ processes$$

Since variance measures the variability from an average or mean. A variance value of zero indicates that all values within a set of numbers are identical; all variances that are non-zero are assigned positive numbers. A large variance indicates that numbers in the set are far from the mean and each other, while a small variance indicates the opposite (http://www.merriam-webster.com/dictionary/variance). The waiting time variance is the measure of how far the waiting time of processes are from the mean of the processes waiting times. This implies that waiting time variance of processes should be minimized (Nong *et al*, 2005).

2.4 CPU Scheduling

The operating system is the most important program needed for starting up and using the hardware. The hardware has different managing tasks each of which is performed by one of the management units of the operating system. The process management, as one of these management units, allocates processor to the processes (Abdulrazaq *et al*, 2014).

A process is a program in execution. The process needs certain resources-including CPU time, memory, files, and I/O devices-to accomplish its task. These resources are either allocated to the process when it is created or allocated to it while it is running. The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization. The objective of time sharing is to switch the CPU among processes so frequently that user can interact with each program while it is running. To meet these objectives, the process

management uses several CPU scheduling algorithms to select an available process (possibly from a set of several available processes) for program execution on the CPU (Abdulrazaq *et al,* 2014).

2.4.1 The importance of priority in CPU Scheduling

A CPU scheduling algorithm should focus on maximizing throughput and CPU utilization and minimizing turnaround time, waiting time and response time. Apart from these parameters there are other factors to consider like number of context switches, fairness etc. Existing scheduling techniques based on priority of the process are not fair and responsive and suffer from the problem of starvation (Saxena and Agarwal 2012). Scheduling the processes is one of the primary job of an operating system. The main goal of the scheduling is the maximization of CPU utilization, throughput and minimization of response time, waiting time and turnaround time. The scheduling is used in the real time applications like routing of data packets in computer networking, controlling traffic in airways, roadways and railways etc. Existing non-preemptive algorithms use only one of the basic parameters such as arrival time (FCFS), user priority (PS) and burst time (SJF) (Rakesh*et al*., 2011).

2.5 Review of Priority Based CPU Scheduling Algorithms

Kleinrock (1970) developed an algorithm known as Selfish Round Robin (SRR) Scheduling Algorithm. This algorithm takes into consideration two queues; the waiting and service queue. The waiting queue is where the customers are assembled on entering the system while the service queue is where the customers are being picked for processor assignment. In both queues, each customer is assigned a priority on entering the queue. For the waiting queue, the priority of the customer increases from zero (upon entry) at a rate α; similarly, when the customer is in

service queue (typically with other customers sharing the service facility simultaneously with him as in a processor shared system) the priority changes at a rate β.

The service queue has a maximum number of customers it can take. The customer with the highest priority is picked for the processor assignment. It is allowed to run until its time quantum expires, and then the next customer with the highest priority in the service queue is allowed to run. In a situation where the customer needs more time, the customer runs for the full length of the time quantum and then it is preempted and then added to the tail of the service queue. New customer can only enter the service queue, if a customer in the service finishes his execution and leaves the system. The customer to be moved from the waiting queue to the service queue is the customer with the highest priority. The priority of customers in the waiting queue α, increases at a faster rate than the priority of customers in the service queue β, who selfishly want to hold on to the processor.

Brinch (1977) proposed an algorithm named Highest Response Ratio Next (HRRN) CPU scheduling algorithm to address the shortcomings of the Shortest Job First (SJF) and Shortest Job Next (SJN) CPU Scheduling algorithms which are starvation and lack of fairness to longer burst time processes. It does this by using the following formula to calculate the Response Ratio of processes

$$ResponseRatio = \frac{WaitingTime + ServiceTime}{ServiceTime} \tag{2.3}$$

It allocates the CPU to the process with the Highest Response Ratio and it also takes care of the problem of determining the next burst time of processes in SJF. Its disadvantage is that it is computationally intensive and its limitation is that it does not address user priority.

Soraj and Roy (n.d) presented a new algorithm called "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice", which was discussed in detail, tested and verified. This algorithm gives the better result compared to Simple Round Robin Scheduling Algorithm. That is, it reduces the average waiting time, context switches, turnaround time, and increases throughput of simple Round Robin Scheduling algorithm. The algorithm is a Priority Driven scheduling algorithm based on burst time of processes. First of all, it arranges the processes according to the execution time/burst time in increasing order that is the smaller the burst time, the higher the priority of the running process. The next idea of this approach is to choose the smart time slice (STS), which is mainly dependant on the number of processes. The smart time slice is equal to the burst time of the mid process, when the number of processes is odd. If the number of processes is even, then the smart time slice (STS) will be the average of the CPU burst of all running processes. Based on the experiments and calculations, the algorithm radically solves the fixed time quantum problem which is considered a challenge for Round Robin scheduling algorithm. This algorithm assumes that all processes arrive at the same time in the ready queue.

Rakesh *et al.*, (2011) proposed a new variant of Round Robin Scheduling algorithm by executing the processes according to a new calculated fit factor F and using the concept of dynamic time quantum. They then compared the performance of their proposed Fittest Job First Dynamic Round Robin Algorithm with the Priority Based Static Round Robin (PBSRR) Algorithm. The experiments they conducted showed that their proposed algorithm performed better than the PBSRR in terms of the number of context switches, average waiting time, and average turnaround time. Generally with every process three factors are associated. These factors are user priority, burst time and arrival time. Above factors play an important role to decide in which

sequence the processes will be executed. Sorting according to the importance of these factors, user priority comes first, then the burst time and at last the arrival time of the processes. In FCFS, SJF and Priority algorithms, only one among the three factors are taken into consideration. If all three factors are mixed up to calculate a new factor that is Fit Factor "f" which will decide the order of execution then average waiting time, average turnaround time and number of context switches will be decreased. But FCFS, SJF and Priority scheduling algorithms are non-preemptive in nature and they can't be used in time sharing systems. So to increase the responsiveness of the system, RR algorithm should be used. Generally in RR algorithm, processes are taken from the ready queue in FCFS manner for execution. But in our algorithm, "f" is calculated for each process. The process having the lowest "f" value will be scheduled first. The two important criteria that decide the early execution of processes are – higher user priority and shorter burst time. As user priority has higher importance than other factors, so it is given a weight age of 60% and burst time is given 40%, assuming that all the processes have same arrival time i.e. arrival time = 0. Let the User Priority = UP, User Priority Weight = UW, Shorter Burst time Priority = SP, Burst time Priority Weight = BW. Then Fit Factor "f" can be calculated as

$$f = UP * UW + SP * BW \quad (2.4)$$

Dynamic time quantum is used in order to overcome the limitations of static RR. To get the optimal time quantum, median of the remaining burst time is taken as the time quantum.

Saxena and Agarwal (2012) designed an algorithm known as Design and Performance Evaluation of Optimum Service Time Concept for Round Robin Algorithm (OSTRR). The

performance of this algorithm was compared with that of Behera *et al* (2010) known as A New Proposed Dynamic Quantum with Re-Adjusted Round Robin (DQRRR) Scheduling Algorithm and Its Performance Analysis. The performance of OSTRR is better than that of DQRRR in terms of minimizing average waiting time, average turnaround time and number of context switches. Generally, with every process, three factors are associated. These factors are user priority, burst time and arrival time. These factors play an important role to decide in which sequence the processes will be executed. Since the algorithm is a priority based system, the concept of Optimum Priority was employed, which combines user defined priority, effect of shorter burst time and effect of arrival time in a way so as to achieve better turnaround time and average waiting time. A weight of 0.5 is assigned to user or system defined priority, a weight of 0.3 to burst time and 0.2 to arrival time. This ensures that user defined priority; burst time; and arrival time get consideration while deciding order of execution of processes. A higher priority process gets a higher number; a shorter process also gets a higher number; and a process that arrived earlier also gets a higher number in the numbering scheme. An Optimum Priority 'OP' is given as:

$$OP = \lceil 0.5 * P + 0.3 * BT + 0.2 * AT \rceil \quad (2.5)$$

Where P is user or system defined priority; BT is priority number assigned according to shorter burst time; and AT is priority number assigned according to early arrival of the process. A time quantum was selected while taking into account the same considerations that is taken while selecting time quantum for RR algorithm. The Optimum service time 'OST' for each process is given by:

$$OST_i = OP_i * q \quad (2.6)$$

Where OST_i is the service time of process with Optimumpriority OP_i. q is the decided time quantum. All the processes are placed in a priority queue. After calculating the service time of each process, CPU is assigned to the process with highest optimum service time. In case of conflict, the process with shorter burst time is given preference. If conflict still persists the process that arrived early is given preference. The process decided executes for a period that is equal to optimum service time of the process OST_i or itsburst time whichever is smallest. In case a process or the setof process with same value of OP, which so ever isapplicable, finish execution after a single round, the value of $OST$is redistributed removing the value of $OST$of finished process and redistributing the value of OSTaccordingly.

2.6 Observations on the Review Literatures Showing the Knowledge Gap Addressed by the Dissertation

Brinch (1977) used an algorithm that is computationally intensive that is its time complexity is of order $O(n^3)$ while this dissertation proposes an algorithm that is of order $O(n^2)$ to address the starvation problem of SJF algorithm. Also, Brinch (1977) does not address the problem of user priority which is addressed by the proposed algorithm in this dissertation.

Rakesh *et al.,* (2011) used only two factors which are burst time and user priority to calculate the Fit factor "f" which they used in scheduling processes in their proposed algorithm while the Efficiency factor used in this dissertation considers three factors which are arrival time, burst time and user priority to schedule processes.

Saxena and Agarwal (2012) after calculating the optimum priority, assigned the CPU to the process with the highest value of the optimum priority while the proposed algorithm in this dissertation allocates the CPU to the process having the least Efficiency factor value. Also, both Rakesh *et al.,* (2011) and Saxena and Agarwal (2012) applied their algorithm to Round Robin

algorithm while the proposed algorithm in this dissertation is applied to priority based CPU scheduling algorithms

2.7 Waiting Time Variance (WTV):

Nong *et al* (2007) in their paper titled "Job scheduling methods for reducing waiting time variance" developed two heuristic job scheduling methods: Balanced Spiral (BS) and Verified Spiral (VS) which made improvements on the algorithm by Eilon and Chowdhury (1977). These scheduling methods were compared with four other scheduling methods (First-In-First-Out (FIFO), Shortest Processing Time (SPT), Eilon and Chowdhury (1977) 1.1 and Eilon and Chowdhury (1977) 1.2 based on waiting time variance. It was observed that Verified Spiral gives the best performance for the scheduling methods but with a very high computation cost. Balanced Spiral produces comparable results, but at less computational cost. In VS, jobs in a pool are rearranged in a manner that the CPU will be assigned to them. The job with the longest service time is placed in the last position, likewise the job with the second longest service time is placed in the last-but-one position. The job with third longest service time is placed in the first position and the job with the shortest service time is placed in the second position. The position to be occupied by the remaining jobs is determined by computing the waiting time variance of the jobs and comparing them with the waiting time variance of the job with shortest service time (i.e. job occupying the second position). The job with the shortest waiting time variance comes first. The computation of the waiting time variance of every job makes this method computational expensive. BS method replaces the calculation of waiting time variance of jobs during the placement selection in order to reduce the computational cost of VS. In BS, the balance of the total servicing time of jobs in the left (L) and right (R) side of the sequence is

maintained, while rearranging the jobs. If the sum of the service times of jobs at the left side is less than the sum of service times of jobs at the right side, the shortest job in the queue is appended at the last position else it is added to the first position. (Nong *et al.,* 2007).

CHAPTER THREE: MATERIALS AND METHOD

3.1 Introduction

This chapter discusses the methodology used in implementing the proposed algorithm.

3.2 Assumption

This dissertation investigates five performance criteria: average waiting time, average turnaround time, average response time, number of context switches and waiting time variance deviation. Because if this criteria are minimized, it proves that starvation has been minimized too which is the main aim of the dissertation. The evaluation is done using a simulation by generating 500 processes randomly with Exponential, Uniform or Normal statistical distributions to generate burst times of the processes. The simulation was done in two different categories; in the first category (i.e. the Shortest Job First (SJF)) consisting of: SJF, the new SJF and Optimum Service Time Round Robin CPU Scheduling algorithms and in the second category (i.e. the Priority Scheduling (PS)) consisting of: PS, the new PS and Optimum Service Time Round Robin CPU Scheduling algorithms to observe these criteria. The simulation environment where all the experiments were performed 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.

3.3 Efficiency Factor

Efficiency factor values for the processes are calculated as follows:

For the first category that is the Shortest Job First (SJF) category, the first process to arrive the arrival queue is scheduled first since it is the only process in the system then and it is allocated the CPU to execute for the period of its burst time. By the time it has finished executing, some

processes will have arrived the arrival queue their efficiency values are calculated by finding the Burst Time Weight (BTW) for all the processes that arrived within this period so that the processes with the least burst time is given a BTW of 1. All other processes are given BTW in ascending order of their burst times. Also each process is given Waiting Time Weight (WW), according to their arrival times so that the first process to arrive is given a WW of 1. Because it has the least waiting time value since it is the first process to arrive the system. All other processes under consideration are given WW of 2,3… until all processes under consideration are given WW. Then the sum of BTW and WW are obtained for each process which is the Efficiency-value for that process and the process with the lowest E-value is scheduled first followed by the process with next lower E-value until all processes that arrive the system have finished executing.

Similarly, for the priority (PS) category, the same method as in the case of SJF is employed but with the inclusion of the Priority Weight (PW), which is calculated based on the priority values of the processes in which the process with the highest priority is given a priority weight of 1 and all other processes are given PW of 2,3…until all processes under consideration are given PW. The sum of WW, BTW and PW are calculated for all the processes under consideration which gives the E-values for the processes they are scheduled based on their E-values such that the process with the least E-value is scheduled first and all other processes are scheduled respectively. This process of determination of E-values and scheduling of processes is performed as explained from the beginning of Section 3.3 until there is no process remaining in the arrival and ready queues respectively.

3.4 The pseudo code of the Proposed Priority Based CPU Scheduling Algorithms

This pseudo code illustrates how the priority based CPU scheduling algorithms work i.e. the Priority and Shortest Job First CPU scheduling algorithms.

Step 1: Start

INPUT: Number of Processes (N), Burst Time (BT) of processes, Arrival Time (AT) of processes, Priority (PRIO) of processes, Queue READY, Queue ARRIVAL

OUTPUT: Number of Context Switches (NCS), Average Waiting Time (AWT), Average Turnaround Time (ATAT), Waiting Time Variance Deviation (WTVD) and Average Response Time (ART)

Step 2: $NCS = 0;$

$$AWT = 0;$$

$ATAT = 0;$

$ART = 0;$

$WTVD = 0;$

$BT =$ Exponential (rate) or Uniform (a, b) or Normal ();

$AT =$ Poisson (rate);

$PRIO =$ Exponential (rate) or Uniform (a, b);

$Last\ Point = 0;$

$$n = N;$$

Step 3: For $i = 1\ to\ n$

$efficiency\ factor_i = 0;$ //*efficiency factor*

END For

Step 4:$WHILE\ (ARRIVAL! = NULL)$

Step 5: $WHILE\ (READY! = NULL)$

Step 6: For $i = 1\ to\ n$

$REQUEST \leftarrow Process_i$//Fill the ready queue according to Arrival Time

//Assign priority weight, burst time weight and arrival weight using the generated values based on the proposed algorithm

END For

Step 7: For $i = 1\ to\ n$

$if\ algorithm == SJF$

$$efficiency\ factor_i = BTW + WW$$

$$else\ if\ algorithm == priority$$

$$efficiency\ factor_i = PW + BTW + WW$$

END if

END For

$$Min = efficiency\ factor[1]$$

Step 8: For ($i = 1\ to\ n - 1$)

If $efficiency\ factor_i < Min$//sort processes according to lowest efficiency factor

$$Min = efficiency\ factor_i$$

$$efficiency\ factor_i = efficiency\ factor_{i+1}$$

$$efficiency\ factor_{i+1} = Min$$

END if

END For

Step 9: For ($i = 1\ ton$) // TAT = Turnaround Time, WT = Waiting Time, RT = Response Time

$$Last\ Point = Last\ Point + BT_i$$

$TAT_i = Last\ Point - AT_i$

$WT_i = TAT_i - BT_i$

$RT = TAT_i - BT_i$

$BT_i = 0$

$NCS_i = NCS_i + 1$

END For

END WHILE

END WHILE

Step 10: Uniform (a, b) { // generate random values using Uniform Distribution

$$a + rand(0,1) * (b - a)$$

} a = upper bound for Uniform Distribution

b = lower bound

Step 11: Poisson (rate) { // generate random values using Poisson Distribution

$$k = 0$$

$$p = 1$$

$$l = e^{-rate}$$

$$do\ \{$$

$$k = k + 1$$

$$p = p * rand(0,1)$$

$$\}while(p \geq l)$$

$$k = k - 1$$

$$return\ k$$

} k= poison parameter

Step 12: Exponential (rate) { // generate random values using Exponential Distribution

$$return\ \frac{-\log(1 - rand(0,1))}{rate}$$

}

Step 13: Normal () { // generate random values using Normal Distribution

$$do\ \{$$

$$x = rand(-1,1)$$

$$y = rand(-1,1)$$

$$r = x^2 + y^2$$

$$\}\ while\ (r \geq 1\ ||r == 0)$$

$$return\ \ (x * \sqrt{-2 * \frac{\log r}{r}})$$

}

Step 14: Calculate OUTPUT parameters

$$AWT = \frac{\sum_{i=1}^{n} WT_i}{n} \quad (3.1)$$

$$ATAT = \frac{\sum_{i=1}^{n} TAT_i}{n} \quad (3.2)$$

$$ART = \frac{\sum_{i=1}^{n} RT_i}{n} \quad (3.3)$$

$$WTV = \frac{\sum_{i=1}^{n} (WT_i - AWT)^2}{n} \quad (3.4)$$

For ($i = 1\ tok$) //k stands for the algorithm under consideration and *opt* is the algorithm with the least waiting time variance.

$$WTVD_k = \frac{WTV_k - WTV_{opt}}{WTV_{opt}} * 100\% \quad (3.5)$$

END For

END

3.5 Proposed Algorithm for Priority CPU Scheduling Algorithm

Generally, in the priority CPU scheduling algorithm, every process is associated with three factors. These factors are priority, burst time and waiting time. These factors play an important role in deciding the sequence that the processes are executed. In First Come First Served (FCFS), SJF and Priority algorithms, only one among these three factors is taken into consideration. If these 3 factors are mixed up to calculate a new factor i.e. Efficiency Factor "E" which is used in deciding the sequence that the processes will be executed, then average waiting time, average turnaround time, average response time and number of context switches will be decreased in the algorithms.

In the proposed algorithm, each factor is assigned a weight based on the level of importance that is processes with high waiting time, short burst time, and high priority are assigned higher weight. An Efficiency Factor "E" is calculated for each process using the formula in equation 3.6:

$$E = PW + \text{BTW} + WW \quad (3.6)$$

where PW is Priority Weight, BTW is Burst time Weight and WW is Waiting time Weight. The process having the lowest E value is scheduled first. The three important criteria that decide the early execution of processes are – higher priority, longer waiting time and shorter burst time. Sexena and Agarwal (2012)

Example

To demonstrate the algorithm in section 3.2, the following example is considered, in which each process has its burst time, arrival time and priority as shown in the Table 3.1. The values were chosen randomly.

Table 3.1: Process Table

PR_ID	AT	BT	PRIO
P1	0	3	10
P2	1	2	6
P3	3	2	4
P4	5	4	3
P5	6	1	7
P6	7	5	11
P7	9	6	8
P8	12	4	9
P9	13	3	1
P10	15	2	5
P11	15	5	2
P12	15	6	13
P13	18	7	12

For the priority scheduling algorithm, the proposed priority algorithm works as follows: Processes are allocated the CPU based on their Efficiency factor values which are calculated as follows: As the first process arrives the arrival queue, it is the only process, so it is scheduled, as it executes for the period of its burst time which is 3 time units. Within this period, the next two processes, that is P2 andP3 must have arrived the arrival queue and hence they are moved to the ready queue for execution. Their efficiency factor is calculated as follows: waiting time weight for P2 is 1 since it has waited for more time than P3 and that ofP3 is 2. Since they have equal

burst time of time units, their burst time weight is 1 each. Priority weight of P3 is 1 since it has a priority value of 4 and priority value of P2 is 2 since it has a priority value of 6. Adding the PW, WW and BTW of P2 and P3 to get their E-values, we get 3 as the E-value for both of them. P3 will be scheduled before P2, since it has a higher priority value of 4 and P2 is scheduled next. After P2 and P3 have finished executing, P4, P5 and P6 have must have arrive the arrival queue and so they will be moved to the ready queue for calculating of their E-values and scheduling them based on it. Waiting weight value for P4 is 1, since it has waited for more time than the other two processes and for P5 is 2 and for P6 is 3 respectively. BTW for P5 is 1 since it has the lowest burst time of 1 time unit and that of P4 is 2 since it has a burst time of 4 time units and that of P6 is 3 since it has a burst time of 5 time units. PW of P4 is 1 since it has a priority of 3 and that of P5 is 2 since it has a priority of 7 and that of P6 is 3 since it has a priority of 11. Calculating the E-values of P4, P5 and P6, we have 4, 5 and 6 respectively. So they are scheduled in the sequence P4, P5, P6. When they have finished executing, the following six processes P7, P8, P9, P10, P11 and P12 will have arrived the arrival queue, so, they will be moved to the ready queue for determination of their E-values scheduling and execution. WW for P7 is 1 since it has waited longer than all the other processes under consideration and that of P8 is 2 and that of P9 is 3 and that of P10, P11 and P12 is 4 since they all arrived at the same time (i.e. 15 time unit). BTW of P10 is 1 since it has the lowest burst time of 2 time units, then that of P9 is 2 which has a burst time of 3 time units, then that of P8 is 3, since it has a burst time of 4 time units and P11 has a BTW of 4 since it has a burst time of 5 time units and that of P7 and P12 is 5, since they both have burst times of 6 time units each. As for the priority weight PW of the six processes under consideration, P9 has priority weight of 1 since it has the highest priority value of 1, then P11 has priority weight of 2 since it has priority value of 2, P10 has priority

weight of 3 since it has priority value of 5 and next, P7 has priority weight of 4 since it has priority value of 8, and P8 has priority weight of 5 since it has priority value of 9 and finally, P12 has priority weight of 6 since it has priority value of 13. When these processes have finished executing, the only process that remains in the arrival queue is P13, which is scheduled and executed for the period of its burst time. These explanations are shown in Table 3.2

Table 3.2: Process Table for Priority CPU Scheduling Algorithm

PR_ID	WW	BTW	PW	Eff. Fac	Ex Sq
P1	1	1	1	3	1
P2	1	1	2	4	2
P3	2	1	1	4	1
P4	1	2	1	4	1
P5	2	1	2	5	2
P6	3	3	3	9	3
P7	1	5	4	10	4
P8	2	3	5	10	5
P9	3	2	1	6	1
P10	4	1	3	8	2
P11	4	4	2	10	3
P12	4	5	6	15	6
P13	1	1	1	3	1

3.6 Proposed Algorithm for Shortest Job First CPU Scheduling Algorithm

In the Short Job First CPU scheduling algorithm, every process is associated with two factors. These factors are burst time and waiting time. These factors in both algorithms play an important role in deciding the sequence that the processes are executed. If these 2 factors are mixed up to calculate a new factor i.e. Efficiency Factor "E" which will be used in deciding the sequence that the processes will be executed, then average waiting time, average turnaround time, average response time and number of context switches will be decreased in the algorithms.

In the proposed algorithm, each factor is assigned a weight based on the level of importance, that is a process with high waiting time will be assigned a higher weight and process with short burst

time will also be assigned a higher weight. An Efficiency Factor "E" is calculated for each process using the formula below:

$$E = BTW + WW \tag{3.7}$$

Where BTW is Burst time Weight and WW is Waiting time Weight. The process having the lowest E value will be scheduled first. The two important criteria that decide the early execution of processes are longer waiting time and shorter burst time.

Example

To demonstrate the algorithm in section 3.3, the following example is considered, in which each process has its burst time and arrival time as shown in the Table 3.3. The values were chosen randomly.

Table 3.3: Process Table

PR_ID	AT	BT
P1	0	3
P2	1	2
P3	3	2
P4	5	4
P5	6	1
P6	7	5
P7	9	6
P8	12	4
P9	13	3
P10	15	2
P11	15	5
P12	15	6
P13	18	7

For the Short Job First CPU scheduling algorithm, the example is illustrated as follows: the first process to arrive the arrival queue P1 is moved to the ready queue and scheduled and it executes for the period of its burst time which is 3 time units. When it finishes executing, the next two

processes i.e. P2 and P3 have arrived the arrival queue so they will be moved to the ready queue for the calculation of their E-values and scheduling accordingly as follows. P2 has WW of 1 since it has waited for a longer period of time than P3 and that of P3 is 2. BTW for both processes is 1 since they have equal burst time of 2 time units. So, the E-value of P2 is 2 and that of P3 is 3, therefore, P2 is scheduled first followed by P3. By the time P2 and P3 have finished executing, the next 3 three processes P4, P5 and P6 have arrived the arrival queue, so they will be moved to the ready queue for calculation of their E-values and scheduling accordingly. The WW of P4 is 1 since it has waited time than all the other two processes and that of P5 is 2 and that of P6 is 3 respectively. The BTW of P5 is 1 since it has the least burst of 1 time unit and that of P4 is 2 since it has a burst time of 4 time units and that of P6 is 3 since it has the burst time 5 time units. The E-value of P4 is 3 and that of P5 is also 3 and that of P6 is 6, therefore, the sequence of execution will be P5 first because it has a lesser burst time than P4, then P4 and finally P6. By the time P4, P5 and P6 have finished executing, the next six processes of P7, P8, P9, P10, P11 and P12 have arrived the arrival queue, so they will be moved to the ready queue and their E-values will be calculated and scheduled accordingly. P7 has a WW of 1 since it has waited for more time than other processes under consideration. P8 has 2, P9 has 3, and P10, P11 and P12 have WW of 4 since they arrived at the same time. P10 has a BTW of 1 since it has the least burst time of 2 time units, P9 has a BTW of 2 since it has a burst time 3 time units, P8 is next with BTW of 3 since it has a burst time of 4 time units, P11 has a BTW of 4 because it has a burst time of 5 time units. P7 and P12 both have a BTW of 5 because they both have burst times of 6 time units. So the E-values of P8, P9 and P10 is 5 each, for P7 is 6, for P11 is 8 and for P12 is 9. So the sequence of execution will be, P10 first because it has the lowest burst time of 2 time units among the processes with which it has the same E-value, the P9 is scheduled since it has

the lowest burst time of 3 and then P8 is scheduled next, then P7 which has an E-value of 4 is scheduled next and P11 with an E-value of 8 is scheduled next and finally P12 is scheduled which has an E-value of 9. After processes P7, P8, P9, P10, P11 and P12 have finished their execution, the only remaining process in the arrival queue is P13 which will be moved to the ready queue and scheduled and it executes for its burst time which is 7 time units. These explanations are shown on Table 3.4.

Table 3.4: Process Table for Shortest Job First CPU Scheduling Algorithm

PR_ID	WW	BTW	Eff. Fac	Ex Sq
P1	1	1	2	1
P2	1	1	2	1
P3	2	1	3	2
P4	1	2	3	1
P5	2	1	3	2
P6	3	3	6	3
P7	1	5	6	4
P8	2	3	5	3
P9	3	2	5	2
P10	4	1	5	1
P11	4	4	8	5
P12	4	5	9	6
P13	1	1	2	1

3.7 Time Complexity of the Proposed Priority Based CPU Scheduling Algorithm

The proposed algorithm would be maintaining all processes that are waiting and ready for execution in arrival and ready queues respectively (Behera *et al*, 2011). Each process insertion in both arrive and ready queues will be achieved in O(1), but process selection (to move to the ready queue) and its deletion would require O(n) time in the arrival queue and process selection (to run next) and its deletion would also require O(n) time in the ready queue, where n is the number of processes in the queue. The proposed algorithm sorts all processes moving from the arrival queue to the ready queue in ascending order of priority (i.e. efficiency factor). This process of sorting will be achieved in $O(n^2)$ using quicksort. When a task arrives, a record for it

can be inserted into the linked list in O(n) time where n is the total number of processes in the priority queue. Therefore, the time complexity of the proposed algorithm is equal to that of quicksort which is $O(n^2)$.

CHAPTER FOUR: RESULTS AND DISCUSSION

4.1 Introduction

This chapter presents the evaluation of the proposed priority based CPU scheduling algorithm. It starts with the assumptions adopted in the system design; followed by the description of how the various scheduling algorithms under study work with the aid of illustrative examples. It also states the system requirement. Finally, the results discussion follows; which is tested using processes that vary between 5 and 500. The results of this implementation are analyzed.

4.2 Illustrative Examples

To demonstrate the previous considerations, we consider this example, in which each process with its burst and arrival time as shown in Table 4.1, where the time quantum used in OSTRR is 10*ms*. The values were chosen randomly, because we wanted to compare the performance of the difference scheduling algorithms under study to see how which of them performs as regards to the criteria under consideration.

Table 4.1: Process Table

PR_ID	AT	BT	PRIO
P1	0	3	10
P2	1	2	6
P3	3	2	4
P4	5	4	3
P5	6	1	7
P6	7	5	11
P7	9	6	8
P8	12	4	9
P9	13	3	1
P10	15	2	5
P11	15	5	2
P12	15	6	13
P13	18	7	12

For evaluation purpose, the formula of Waiting Time (WT) (i.e. (4.1)) shall be used in calculating the Average Waiting Time (AWT)(i.e. (4.2)) for each scheduling algorithm.

$$Waiting\ Time = Time\ first\ scheduled - Arrival\ Time \quad (4.1)$$

$$Average\ Waiting\ Time = \frac{Sum\ of\ all\ processes\ Waiting\ Time}{Number\ of\ processes} \quad (4.2)$$

The Turnaround Time (TAT) (i.e. (4.3)) shall also be used in calculating the Average Turnaround Time (ATAT) (i.e. (4.4)) for each scheduling algorithm.

$$Turnaround\ Time = Time\ of\ process\ completion - Arrival\ Time \quad (4.3)$$

$$Average\ Turnaround\ Time = \frac{Sum\ of\ all\ processes\ Turnaround\ Time}{Number\ of\ processes} \quad (4.4)$$

And the Response Time (RT) (i.e. 4.5) shall also be used in calculating the Average Response Time (ART) (that is 4.6) for each scheduling algorithm as follows (Silberschatz *et al.*, 2006):

$$ResponseTime = Time\ of\ process's\ first\ response - Arrival\ Time \quad (4.5)$$

$$Average\ Response\ Time = \frac{Sum\ of\ all\ processes\ Response\ Time}{Number\ of\ processes} \quad (4.6)$$

4.2.1 Shortest Job First (SJF)

Figure 4.1 shows the Gantt chart of Shortest Job First CPU scheduling algorithm

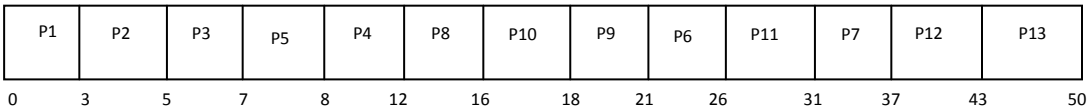


Figure4. 1: Gantt chart representation of SJF

Waiting Time

P1: (0-0) = 0, P2: (3-1) =2, P3: (5-3) = 2, P4: (8-5) = 3, P5: (7-6) = 1, P6: (21-7) = 14, P7: (31-9) = 22, P8: (12-12) = 0, P9: (18-13) = 5, P10: (16-15) = 1, P11: (26-15) = 11, P12: (37-15) = 22, P13: (43-18) = 25.

Average Waiting Time

$$AWT = \frac{(0 + 2 + 2 + 3 + 1 + 14 + 22 + 0 + 5 + 1 + 11 + 22 + 25)}{13} = \frac{108}{13} = 8.31$$

Turnaround Time

P1: (3-0) = 3, P2: (5-1) =4, P3: (7-3) = 4, P4: (12-5) = 12, P5: (8-6) = 2, P6: (26-7) = 19, P7: (37-9) = 28, P8: (16-12) = 0, P9: (21-13) = 18, P10: (18-15) = 3, P11: (31-15) = 16, P12: (43-15) = 28, P13: (50-18) = 32.

Average Turnaround Time

$$ATAT = \frac{(3 + 4 + 4 + 12 + 2 + 19 + 28 + 4 + 18 + 3 + 16 + 28 + 32)}{13} = \frac{173}{13} = 13.31$$

Response Time

P1: (0-0) = 0, P2: (3-1) =2, P3: (5-3) = 2, P4: (8-5) = 3, P5: (7-6) = 1, P6: (21-7) = 14, P7: (31-9) = 22, P8: (12-12) = 0, P9: (18-13) = 5, P10: (16-15) = 1, P11: (26-15) = 11, P12: (37-15) = 22, P13: (43-18) = 25.

Average Response Time

$$ART = \frac{(0 + 2 + 2 + 3 + 1 + 14 + 22 + 0 + 5 + 1 + 11 + 22 + 25)}{13} = \frac{108}{13} = 8.31$$

Number of Context Switches = 0

4.2.2 Priority Scheduling (PS)

Figure 4.2 shows the Gantt chart of Priority CPU scheduling algorithm

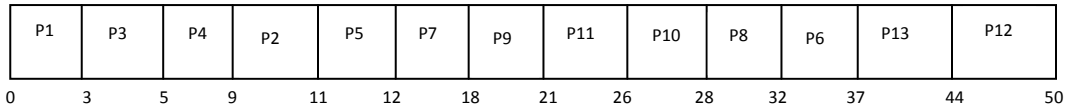


Figure4. 2: Gantt chart representation of Priority CPU Scheduling Algorithm

Waiting Time

P1: (0-0) = 0, P2: (9-1) =8, P3: (3-3) = 0, P4: (5-5) = 0, P5: (11-6) = 5, P6: (32-7) = 25, P7: (12-9) = 3, P8: (28-12) = 16, P9: (18-13) = 5, P10: (26-15) = 11, P11: (21-15) = 6, P12: (44-15) = 29, P13: (37-18) = 19.

Average Waiting Time

$$AWT = \frac{(0 + 8 + 0 + 0 + 5 + 25 + 3 + 16 + 5 + 11 + 6 + 29 + 19)}{13} = \frac{127}{13} = 9.77$$

Turnaround Time

P1: (3-0) = 3, P2: (11-1) =10, P3: (5-3) = 2, P4: (9-5) = 4, P5: (12-6) = 6, P6: (39-7) = 32, P7: (21-9) = 12, P8: (32-12) = 20, P9: (21-13) = 8, P10: (28-15) = 13, P11: (26-15) = 11, P12: (50-15) = 35, P13: (44-18) = 26.

Average Turnaround Time

$$ATAT = \frac{(3 + 10 + 2 + 4 + 6 + 30 + 9 + 20 + 8 + 13 + 11 + 35 + 26)}{13} = \frac{177}{13} = 13.62$$

Response Time

P1: (0-0) = 0, P2: (9-1) =8, P3: (3-3) = 0, P4: (5-5) = 0, P5: (11-6) = 5, P6: (32-7) = 25, P7: (12-9) = 3, P8: (28-12) = 16, P9: (18-13) = 5, P10: (26-15) = 11, P11: (21-15) = 6, P12: (44-15) = 29, P13: (37-18) = 19.

Average Response Time

$$ART = \frac{(0 + 8 + 0 + 0 + 5 + 25 + 3 + 16 + 5 + 11 + 6 + 29 + 19)}{13} = \frac{127}{13} = 9.77$$

Number of Context Switches = 0

4.2.3 Proposed Shortest Job First

Figure 4.3 shows the Gantt chart of Proposed Shortest Job First CPU scheduling algorithm

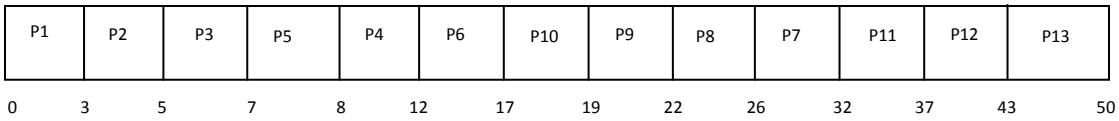


Figure4. 3: Gantt chart representation of Proposed Shortest Job First

Waiting Time

P1: (0-0) = 0, P2: (3-1) =2, P3: (5-3) = 2, P4: (8-5) = 3, P5: (7-6) = 1, P6: (12-7) = 5, P7: (26-9) = 17, P8: (22-12) = 10, P9: (19-13) = 6, P10: (17-15) = 2, P11: (32-15) = 17, P12: (37-15) = 22, P13: (43-18) = 25.

Average Waiting Time

$$AWT = \frac{(0 + 2 + 2 + 3 + 1 + 5 + 17 + 10 + 6 + 2 + 17 + 22 + 25)}{13} = \frac{112}{13} = 8.62$$

Turnaround Time

P1: (3-0) = 3, P2: (5-1) =4, P3: (9-3) = 6, P4: (14-5) = 9, P5: (8-6) = 2, P6: (17-7) = 10, P7: (34-9) = 23, P8: (26-12) = 14, P9: (22-13) = 9, P10: (19-15) = 4, P11: (37-15) = 22, P12: (43-15) = 28, P13: (50-18) = 32.

Average Turnaround Time

$$AWT = \frac{(3+4+6+9+2+10+23+14+9+4+22+28+32)}{13} = \frac{162}{13} = 12.46$$

Response Time

P1: (0-0) = 0, P2: (3-1) =2, P3: (5-3) = 2, P4: (8-5) = 3, P5: (7-6) = 1, P6: (12-7) = 5, P7: (26-9) = 17, P8: (22-12) = 10, P9: (19-13) = 6, P10: (17-15) = 2, P11: (32-15) = 17, P12: (37-15) = 22, P13: (43-18) = 25.

Average Response Time

$$ART = \frac{(0+2+2+3+1+5+17+10+6+2+17+22+25)}{13} = \frac{112}{13} = 8.62$$

Number of Context Switches = 0

4.2.4 Proposed Priority Scheduling

Figure 4.4 shows the Gantt chart of Proposed Priority CPU scheduling algorithm

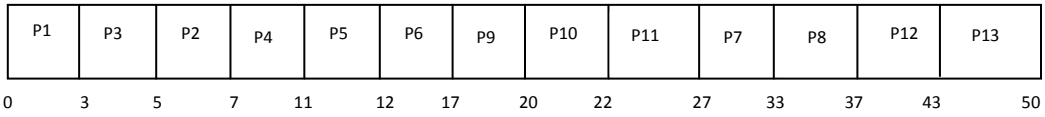


Figure4. 4: Gantt chart representation of Proposed Priority Scheduling

Waiting Time

P1: (0-0) = 0, P2: (4-1) =3, P3: (4-3) = 1, P4: (8-5) = 6, P5: (11-6) = 5, P6: (12-7) = 5, P7: (27-9) = 18, P8: (33-12) = 21, P9: (17-13) = 4, P10: (20-15) = 5, P11: (22-15) = 7, P12: (37-15) = 22, P13: (43-18) = 25.

Average Waiting Time

$$AWT = \frac{(0+3+1+3+5+5+18+21+4+5+7+22+25)}{13} = \frac{119}{13} = 9.15$$

Turnaround Time

P1: (3-0) = 3, P2: (6-1) =5, P3: (8-3) = 5, P4: (14-5) = 9, P5: (12-6) = 6, P6: (17-7) = 10, P7: (33-9) = 24, P8: (37-12) = 25, P9: (20-13) = 7, P10: (22-15) = 7, P11: (27-15) = 12, P12: (43-15) = 28, P13: (50-18) = 32.

Average Turnaround Time

$$ATAT = \frac{(3+4+6+12+2+19+28+4+18+3+16+28+32)}{13} = \frac{169}{13} = 13$$

Response Time

P1: (0-0) = 0, P2: (4-1) =3, P3: (4-3) = 1, P4: (8-5) = 6, P5: (11-6) = 5, P6: (12-7) = 5, P7: (27-9) = 18, P8: (33-12) = 21, P9: (17-13) = 4, P10: (20-15) = 5, P11: (22-15) = 7, P12: (37-15) = 22, P13: (43-18) = 25.

Average Response Time

$$ART == \frac{(0+3+1+3+5+5+18+21+4+5+7+22+25)}{13} = \frac{119}{13} = 9.15$$

Number of Context Switches = 0

4.2.5 OSTRR

Figure 4.5 shows the Gantt chart of OSTRR CPU scheduling algorithm

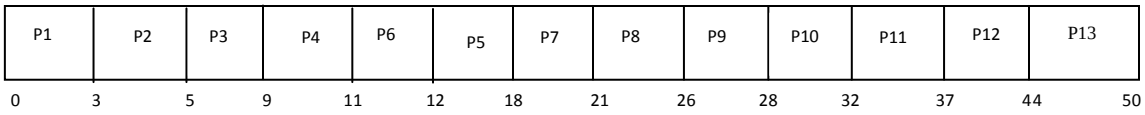


Figure4. 5: Gantt chart representation of OSTRR

Waiting Time

P1: (0-0) = 0, P2: (2-1) =1, P3: (5-3) = 2, P4: (6-5) = 1, P5: (8-6) = 2, P6: (8-7) = 1, P7: (11-9) = 2, P8: (26-12) = 14, P9: (27-13) = 14, P10: (29-15) = 14, P11: (37-15) = 22, P12: (35-15) = 20, P13: (48-18) = 30.

Average Waiting Time

$$AWT = \frac{(0 + 1 + 2 + 1 + 2 + 1 + 2 + 14 + 14 + 14 + 22 + 20 + 30)}{13} = \frac{123}{13} = 9.46$$

Turnaround Time

P1: (3-0) = 3, P2: (4-1) =3, P3: (7-3) = 4, P4: (10-5) = 5, P5: (9-6) = 3, P6: (13-7) = 6, P7: (17-9) = 8, P8: (30-12) = 18, P9: (30-13) = 17, P10: (31-15) = 16, P11: (42-15) = 27, P12: (41-15) = 26, P13: (55-18) = 37.

Average Turnaround Time

$$ATAT = \frac{(3 + 3 + 4 + 5 + 3 + 6 + 8 + 18 + 17 + 16 + 27 + 26 + 37)}{13} = \frac{173}{13} = 13.31$$

Response Time

P1: (0-0) = 0, P2: (2-1) =1, P3: (5-3) = 2, P4: (6-5) = 1, P5: (8-6) = 2, P6: (8-7) = 1, P7: (11-9) = 2, P8: (26-12) = 14, P9: (27-13) = 14, P10: (29-15) = 14, P11: (37-15) = 22, P12: (35-15) = 20, P13: (48-18) = 30.

Average Response Time

$$ART = \frac{(0 + 1 + 2 + 1 + 2 + 1 + 2 + 14 + 14 + 14 + 22 + 20 + 30)}{13} = \frac{123}{13} = 9.46$$

Number of Context Switches = 0

Table 4.2: Comparative Table

Algorithms	AWT	ATAT	ART	NCS	WTVD (%)	WTVD (%)
SJF	8.31	13.31	8.31	0	16.76	
Priority	9.77	13.62	9.77	0		18.93
Pro. SJF	8.62	12.46	8.62	0	0	
Pro. Priority	9.15	13	9.15	0		0
OSTRR	9.46	13.21	9.46	0	35.15	29.62

Table 4.2 shows the comparative results of the algorithms under study. In the SJF category, the proposed algorithm produced the best result followed by the traditional SJF algorithm and the OSTRR. In the priority category the proposed algorithm produced the best result followed by the traditional priority algorithm followed by OSTRR.

4.3 System Requirements

The system requirements for this dissertation are as follows:

4.3.1 Experimental Setup

Hardware

- TOSHIBA Pentium® Dual-Core CPU T4400 with RAM of 4GB running at 2.2GHz

Software

- NetBeans IDE 6.9.1 version and JDK1.7

A process generator routine to generate the process sets was built. Each process in the process set is a tuple: <(process_*id*, *arrival_time*, *CPU_time*, *priority*)>. The process arrival was modeled as a Poisson random process. Hence, the inter-arrival times are Poisson distributed. A process arrival generator was developed to take care of the random arrival of different processes to the system. The generator produces the inter-arrival times utilizing some specific mean (arrival intensity) of the distribution function.

Let $F(x)$ be a Poisson process with a mean $\alpha>0$. So, using the Inverse Transformation method:

$$F(x) = \frac{e^{-\alpha}\alpha^{x}}{x!} \quad (4.7)$$

One important property of the Poisson distribution is that the mean and variance are both equal to α, that is,(Jerry *et al*, 2005)

$$E(X) = \alpha = V(X) \quad (4.8)$$

Burst time (that is the *CPU_time*) was generated randomly using Normal, Uniform or Exponential Distributions. FOR Normal Distribution, a mean of 50.5 ms and a standard Deviation of 28.6 ms where used to generate the values randomly. For Uniform Distribution, the range of values of 1 to 100 ms for burst time of processes was used to generate the values randomly and for Exponential Distribution, an arrival rate (λ) of processes of 0.02 per ms was used to generate the value randomly. Process priority (i.e. *priority*) was generated using either uniform or exponential distribution. A process burst time generator was developed to take care of the random burst time of different processes in the system and also a process priority generator was developed to take care of the random priority of different processes in the system.

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

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad (4.9)$$

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:

$$f(x) = \frac{1}{b-a} \quad (4.10)$$

Where $a \leq x \leq b$

And the cumulative density function (cdf) is given by:

$$F(x) = \frac{x-a}{b-a} = R \qquad (4.11)$$

Or

$$x = a + (b-a)R \qquad (4.12)$$

Which is a reasonable guess for generating X and R is always a random number on [0, 1].

Its mean and variance is given by(Jerry *et al*, 2005):

$$E(X) = \frac{a+b}{2} and\ V(X) = \frac{(b-a)^2}{12} \qquad (4.13)$$

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:

$$f(x) = \lambda e^{-\lambda x} \qquad (4.14)$$

It has mean and variance is given by(Jerry *et al*, 2005):

$$E(X) = \frac{1}{\lambda} and\ V(X) = \frac{1}{\lambda^2} \qquad (4.15)$$

4.4 Results and Discussion

Figure 4.6 shows the system interface through which the user can effectively communicate with the system. It also shows some results of computing 500 processes with burst times that are uniformly distributed with mean of 50.5*ms* and the time quantum of 10*ms* used by OSTRR.

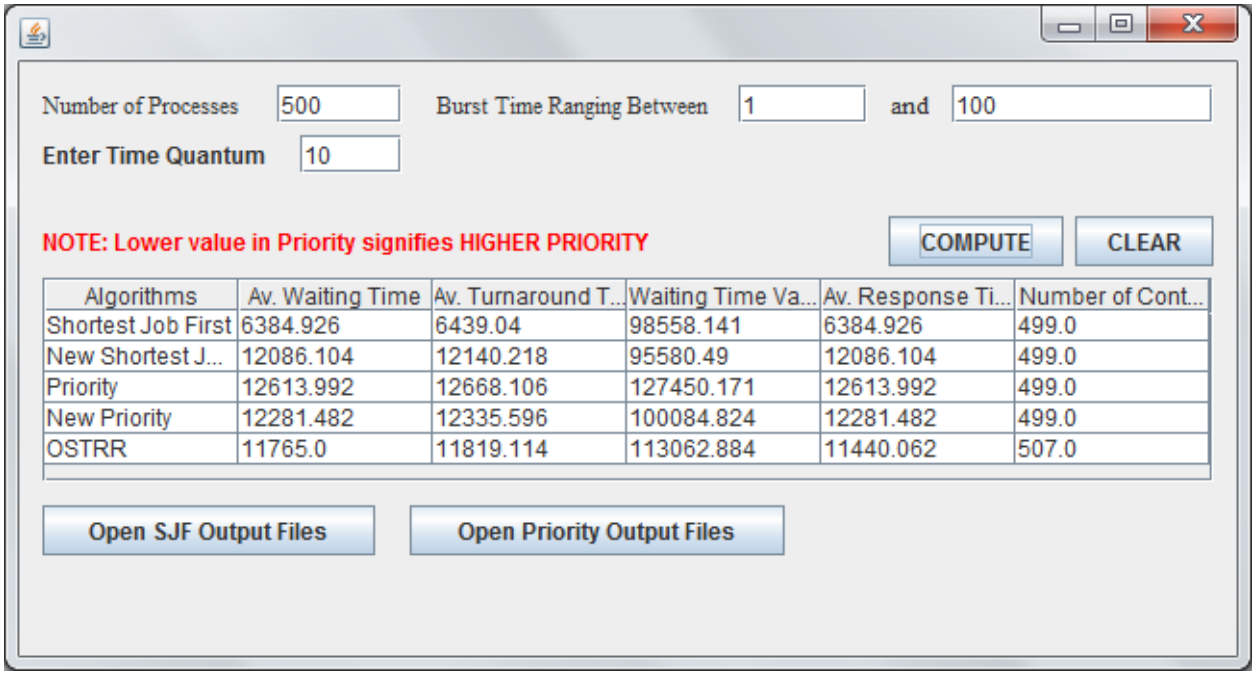


Figure 4. 6: System Interface with some Process Execution in Progress

4.4.1 Results obtained using Normal Distribution to generate burst time

The following shows the relationships between the CPU scheduling algorithms under study using 500 processes generated using normal distribution with a mean of 50.5*ms*, standard deviation of 28.6*ms* and time quantum of 10*ms* used by OSTRR.

4.4.1.1 First category

AWT - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	57.8	57.8	76.2
10	141.0	149.1	203.4
15	222.5	243.7	357.0
20	295.4	328.4	422.6
25	376.9	459.0	504.8
30	450.6	576.1	608.8
35	546.8	693.7	727.5
40	608.0	786.0	826.2
45	677.6	885.5	929.6
50	809.1	1011.0	1036.7
55	854.7	1102.2	1207.3
60	942.1	1202.1	1366.0
65	1024.1	1302.0	1438.2
70	1092.8	1399.1	1585.0
75	1156.2	1502.3	1655.6
80	1261.6	1615.6	1901.8
85	1356.9	1727.6	2002.4
90	1436.6	1823.1	1975.7
95	1528.6	1927.1	2086.3
100	1681.2	2063.6	2237.3
110	1850.8	2283.0	2402.0
120	2090.9	2532.9	2678.1
130	2196.4	2723.3	2903.8
140	2293.2	2927.0	3118.2
150	2467.8	3169.8	3259.6
160	2590.2	3388.7	3431.0
170	2712.3	3619.6	3573.1
180	2805.4	3825.5	3775.5
190	2863.6	4026.8	3987.8
200	3078.4	4267.3	4217.0
220	3385.2	4708.8	4547.7
240	3705.0	5139.5	4919.1
260	4047.0	5596.1	5406.8
280	4332.0	6045.9	5848.1
300	4615.3	6502.0	6347.9
320	4925.4	6996.7	6783.8
340	5161.6	7476.6	7175.1
360	5551.3	7958.0	7670.0
380	5869.2	8425.4	8157.3
400	6149.7	8879.2	8563.9
425	6558.8	9449.4	9085.3
450	6896.7	10020.0	9707.7
475	7208.0	10592.8	10249.8
500	7537.6	11161.9	10797.6

ATAT - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	99.0	99.0	117.4
10	189.6	197.7	252.0
15	271.7	292.9	406.2
20	345.9	379.3	473.1
25	427.6	510.1	555.6
30	500.2	626.0	658.4
35	597.4	744.5	778.1
40	657.9	836.1	876.0
45	727.4	935.4	979.3
50	861.3	1063.3	1088.9
55	905.5	1151.9	1258.1
60	993.4	1253.3	1417.3
65	1075.6	1352.6	1489.7
70	1144.3	1450.7	1636.5
75	1207.0	1552.3	1706.5
80	1313.3	1667.4	1953.4
85	1409.0	1778.9	2054.5
90	1488.9	1874.7	2028.0
95	1581.1	1979.7	2138.8
100	1735.1	2117.0	2291.3
110	1904.6	2336.9	2455.8
120	2145.5	2587.0	2732.8
130	2249.9	2776.9	2957.2
140	2345.6	2979.5	3170.6
150	2520.3	3222.2	3312.2
160	2642.3	3440.8	3483.1
170	2763.8	3670.7	3624.6
180	2856.0	3876.1	3826.1
190	2913.5	4076.3	4037.6
200	3128.9	4317.5	4267.5
220	3435.7	4759.1	4598.3
240	3755.6	5190.2	4969.8
260	4097.7	5646.9	5457.5
280	4382.7	6096.6	5898.8
300	4665.7	6552.3	6398.3
320	4975.6	7046.9	6834.0
340	5211.4	7526.5	7225.0
360	5601.7	8008.4	7720.3
380	5919.5	8475.7	8207.6
400	6199.9	8929.4	8614.2
425	6609.1	9499.7	9135.6
450	6946.6	10070.0	9757.6
475	7257.6	10642.4	10299.4
500	7586.9	11211.2	10846.9

Figure4. 7: Data for Average Waiting Time and Average Turnaround Time using Normal Distribution for Shortest Job First

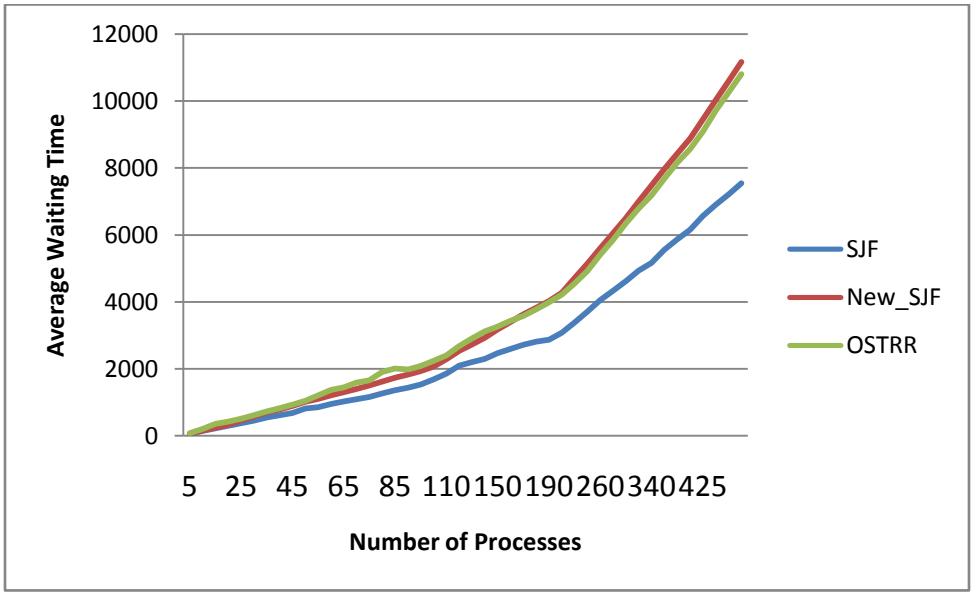


Figure4. 8: Graph of Average Waiting Time using Normal Distribution

Figure 4.8 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm has almost the same performance with OSTRR in terms of minimizing AWT.

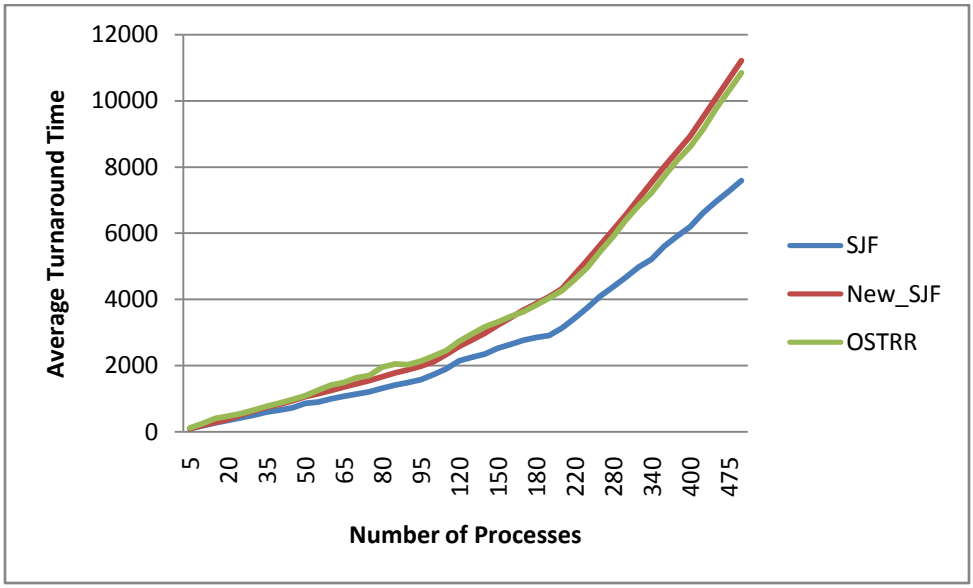
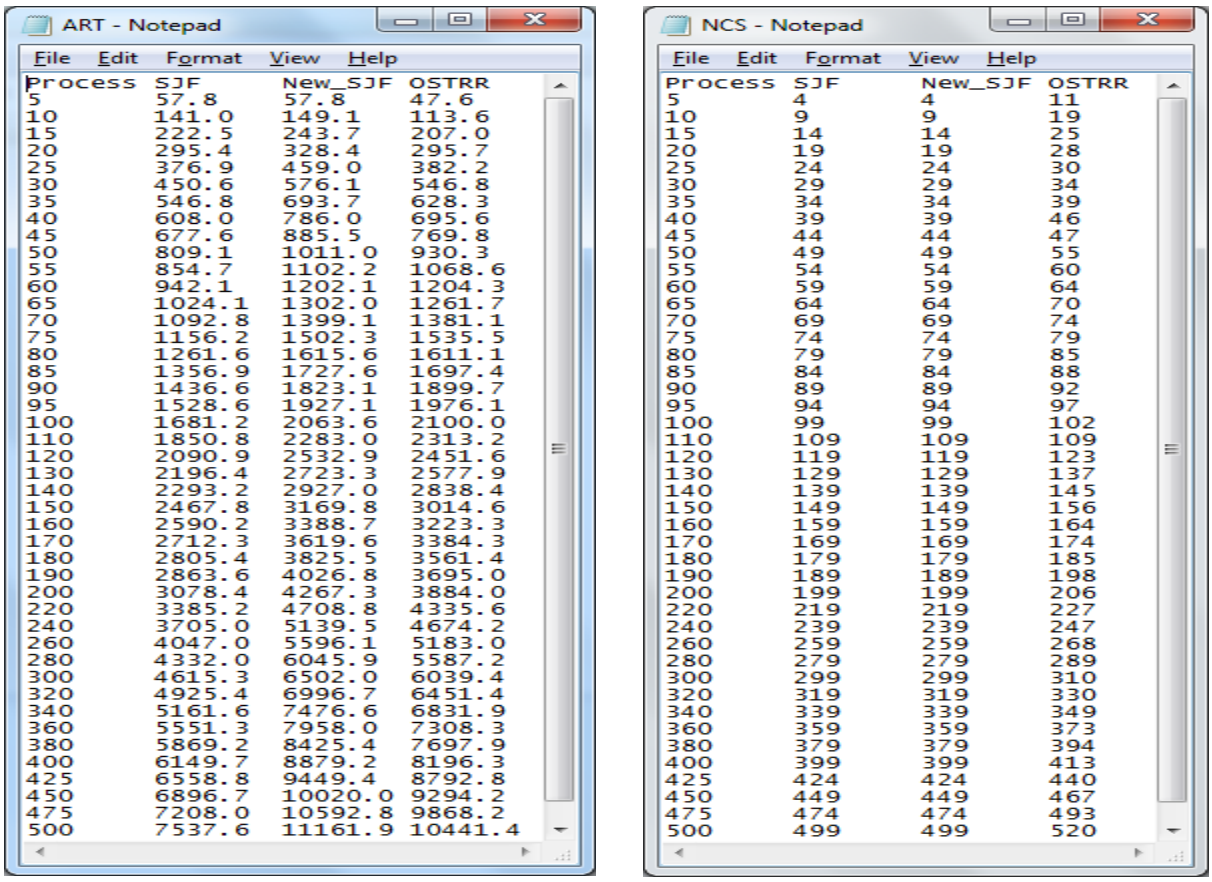


Figure4. 9: Graph of Average Turnaround Time using Normal Distribution

Figure 4.9 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm has almost the same performance with OSTRR in terms of minimizing ATAT.



ART - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	57.8	57.8	47.6
10	141.0	149.1	113.6
15	222.5	243.7	207.0
20	295.4	328.4	295.7
25	376.9	459.0	382.2
30	450.6	576.1	546.8
35	546.8	693.7	628.3
40	608.0	786.0	695.6
45	677.6	885.5	769.8
50	809.1	1011.0	930.3
55	854.7	1102.2	1068.6
60	942.1	1202.1	1204.3
65	1024.1	1302.0	1261.7
70	1092.8	1399.1	1381.1
75	1156.2	1502.3	1535.5
80	1261.6	1615.6	1611.1
85	1356.9	1727.6	1697.4
90	1436.6	1823.1	1899.7
95	1528.6	1927.1	1976.1
100	1681.2	2063.6	2100.0
110	1850.8	2283.0	2313.2
120	2090.9	2532.9	2451.6
130	2196.4	2723.3	2577.9
140	2293.2	2927.0	2838.4
150	2467.8	3169.8	3014.6
160	2590.2	3388.7	3223.3
170	2712.3	3619.6	3384.3
180	2805.4	3825.5	3561.4
190	2863.6	4026.8	3695.0
200	3078.4	4267.3	3884.0
220	3385.2	4708.8	4335.6
240	3705.0	5139.5	4674.2
260	4047.0	5596.1	5183.0
280	4332.0	6045.9	5587.2
300	4615.3	6502.0	6039.4
320	4925.4	6996.7	6451.4
340	5161.6	7476.6	6831.9
360	5551.3	7958.0	7308.3
380	5869.2	8425.4	7697.9
400	6149.7	8879.2	8196.3
425	6558.8	9449.4	8792.8
450	6896.7	10020.0	9294.2
475	7208.0	10592.8	9868.2
500	7537.6	11161.9	10441.4

NCS - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	4	4	11
10	9	9	19
15	14	14	25
20	19	19	28
25	24	24	30
30	29	29	34
35	34	34	39
40	39	39	46
45	44	44	47
50	49	49	55
55	54	54	60
60	59	59	64
65	64	64	70
70	69	69	74
75	74	74	79
80	79	79	85
85	84	84	88
90	89	89	92
95	94	94	97
100	99	99	102
110	109	109	109
120	119	119	123
130	129	129	137
140	139	139	145
150	149	149	156
160	159	159	164
170	169	169	174
180	179	179	185
190	189	189	198
200	199	199	206
220	219	219	227
240	239	239	247
260	259	259	268
280	279	279	289
300	299	299	310
320	319	319	330
340	339	339	349
360	359	359	373
380	379	379	394
400	399	399	413
425	424	424	440
450	449	449	467
475	474	474	493
500	499	499	520

Figure4. 10: Data for Average Response Time and Number of Context Switches using Normal Distribution for Shortest Job First

Figure 4.11 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that the traditional SJF algorithm is better than the proposed algorithm while OSTRR is better than the proposed algorithm in terms of minimizing ART. Figure 4.12 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the traditional algorithm and the proposed algorithm produce the same number of context switches which is better than that of OSTRR.

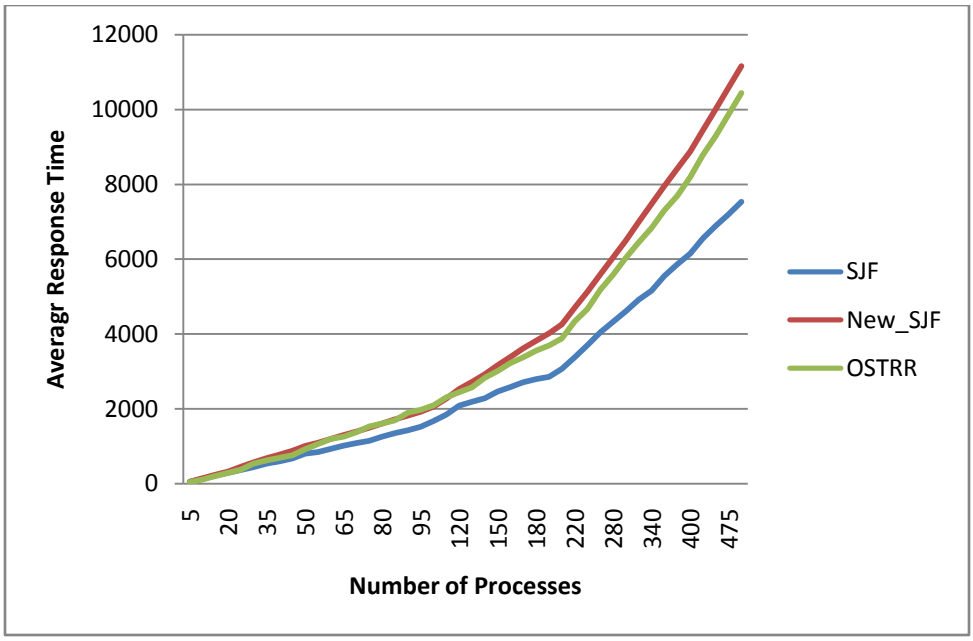


Figure4. 11: Graph of Average Response Time using Normal Distribution

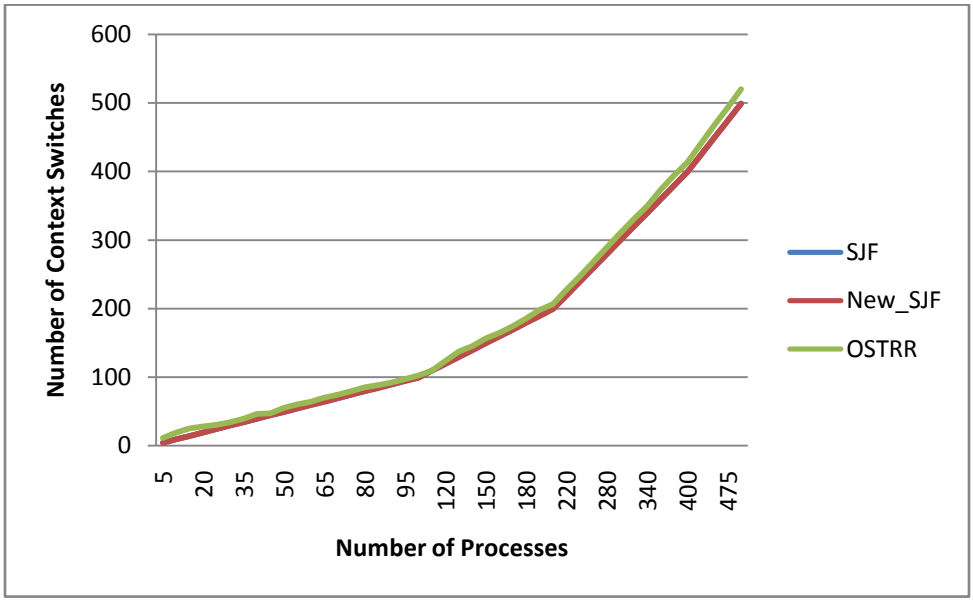
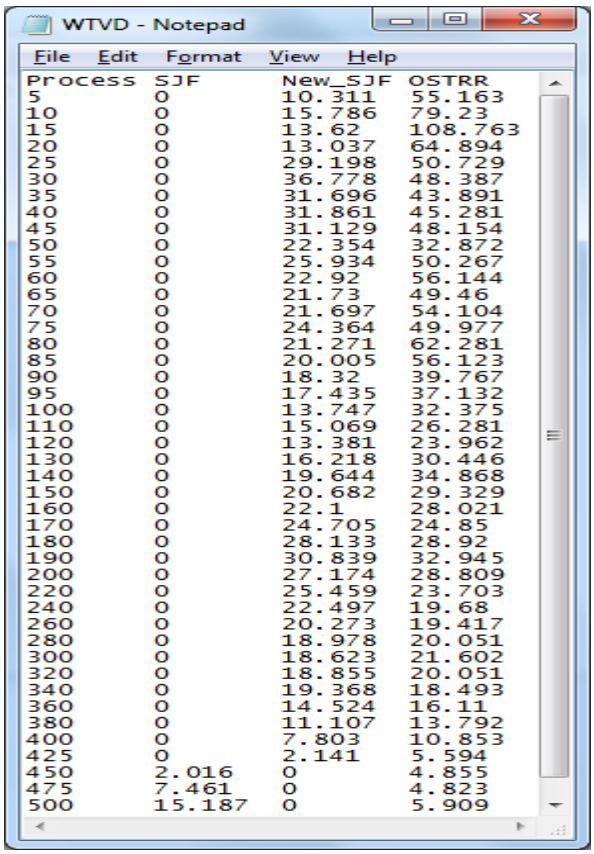


Figure4. 12: Graph of Number of Context Switches using Normal Distribution



WTVD - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	0	10.311	55.163
10	0	15.786	79.23
15	0	13.62	108.763
20	0	13.037	64.894
25	0	29.198	50.729
30	0	36.778	48.387
35	0	31.696	43.891
40	0	31.861	45.281
45	0	31.129	48.154
50	0	22.354	32.872
55	0	25.934	50.267
60	0	22.92	56.144
65	0	21.73	49.46
70	0	21.697	54.104
75	0	24.364	49.977
80	0	21.271	62.281
85	0	20.005	56.123
90	0	18.32	39.767
95	0	17.435	37.132
100	0	13.747	32.375
110	0	15.069	26.281
120	0	13.381	23.962
130	0	16.218	30.446
140	0	19.644	34.868
150	0	20.682	29.329
160	0	22.1	28.021
170	0	24.705	24.85
180	0	28.133	28.92
190	0	30.839	32.945
200	0	27.174	28.809
220	0	25.459	23.703
240	0	22.497	19.68
260	0	20.273	19.417
280	0	18.978	20.051
300	0	18.623	21.602
320	0	18.855	20.051
340	0	19.368	18.493
360	0	14.524	16.11
380	0	11.107	13.792
400	0	7.803	10.853
425	0	2.141	5.594
450	2.016	0	4.855
475	7.461	0	4.823
500	15.187	0	5.909

Figure4. 13: Data for Waiting Time Variance Deviation using Normal Distribution for Shortest Job First

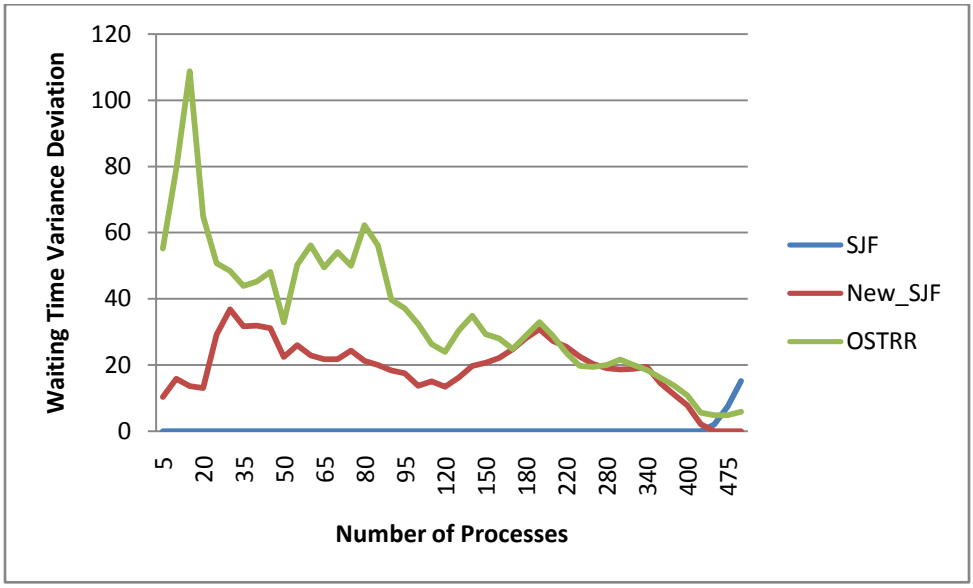


Figure4. 14: Graph of Waiting Time Variance Deviation

Figure 4.14 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing WTVD. But, as number of processes exceeds 450, the proposed algorithm produced the better result.

From the results of the two scheduling algorithms (SJF and OSTRR) compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced better Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared with the proposed new SJF, but as number of processes exceeds 450, the proposed algorithm produced the better result. But they both produced the same Number of Context Switches (NCS). The new proposed SJF produced better results compared with OSTRR with respect to Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD).

4.4.1.2 Second category

AWT - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	69.6	59.4	76.2
10	172.9	157.2	203.4
15	269.9	247.2	357.0
20	406.7	336.5	422.6
25	532.1	472.9	504.8
30	635.5	579.2	608.8
35	765.9	707.5	727.5
40	857.2	795.9	826.2
45	988.0	911.2	929.6
50	1161.2	1045.5	1036.7
55	1229.4	1128.3	1207.3
60	1319.7	1228.3	1366.0
65	1420.9	1333.7	1438.2
70	1547.1	1425.8	1585.0
75	1620.8	1518.6	1655.6
80	1734.1	1628.2	1901.8
85	1870.6	1742.4	2002.4
90	2005.8	1849.0	1975.7
95	2141.1	1942.7	2086.3
100	2320.2	2080.6	2237.3
110	2623.8	2316.6	2402.0
120	2923.7	2559.9	2678.1
130	3104.4	2750.2	2903.8
140	3303.6	2959.8	3118.2
150	3530.8	3195.0	3259.6
160	3747.2	3425.9	3431.0
170	3971.3	3675.7	3573.1
180	4147.6	3879.4	3775.5
190	4302.9	4060.7	3987.8
200	4592.0	4297.9	4217.0
220	5035.3	4742.6	4547.7
240	5558.1	5183.0	4919.1
260	6066.0	5636.0	5406.8
280	6522.8	6099.5	5848.1
300	6954.0	6547.5	6347.9
320	7384.9	7042.8	6783.8
340	7839.5	7525.9	7175.1
360	8344.7	8004.8	7670.0
380	8766.4	8464.2	8157.3
400	9177.0	8917.6	8563.9
425	9823.0	9495.2	9085.3
450	10277.7	10056.1	9707.7
475	10872.8	10637.6	10249.8
500	11315.4	11192.1	10797.6

ATAT - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	100.6	100.6	117.4
10	221.5	201.5	252.0
15	319.1	293.5	406.2
20	455.0	388.8	473.1
25	583.2	525.2	555.6
30	685.1	630.0	658.4
35	816.5	759.1	778.1
40	906.0	846.7	876.0
45	1037.9	961.4	979.3
50	1213.4	1098.3	1088.9
55	1280.4	1178.4	1258.1
60	1370.5	1280.8	1417.3
65	1472.0	1385.7	1489.7
70	1598.5	1478.0	1636.5
75	1671.6	1569.1	1706.5
80	1785.4	1679.1	1953.4
85	1922.2	1795.0	2054.5
90	2057.7	1901.5	2028.0
95	2193.1	1996.0	2138.8
100	2373.9	2134.2	2291.3
110	2677.3	2370.1	2455.8
120	2978.1	2614.9	2732.8
130	3157.7	2804.0	2957.2
140	3355.9	3013.0	3170.6
150	3583.9	3247.0	3312.2
160	3799.5	3478.3	3483.1
170	4023.0	3727.3	3624.6
180	4197.8	3930.2	3826.1
190	4352.4	4110.8	4037.6
200	4642.2	4348.6	4267.5
220	5085.6	4793.3	4598.3
240	5608.9	5233.9	4969.8
260	6117.0	5686.7	5457.5
280	6573.6	6150.3	5898.8
300	7004.5	6597.7	6398.3
320	7435.2	7092.8	6834.0
340	7889.4	7575.5	7225.0
360	8395.0	8055.0	7720.3
380	8816.9	8514.4	8207.6
400	9227.4	8967.6	8614.2
425	9873.6	9545.4	9135.6
450	10327.9	10105.8	9757.6
475	10922.7	10687.1	10299.4
500	11364.9	11241.3	10846.9

Figure4. 15: Data for Average Waiting Time and Average Turnaround Time using Normal Distribution for Priority

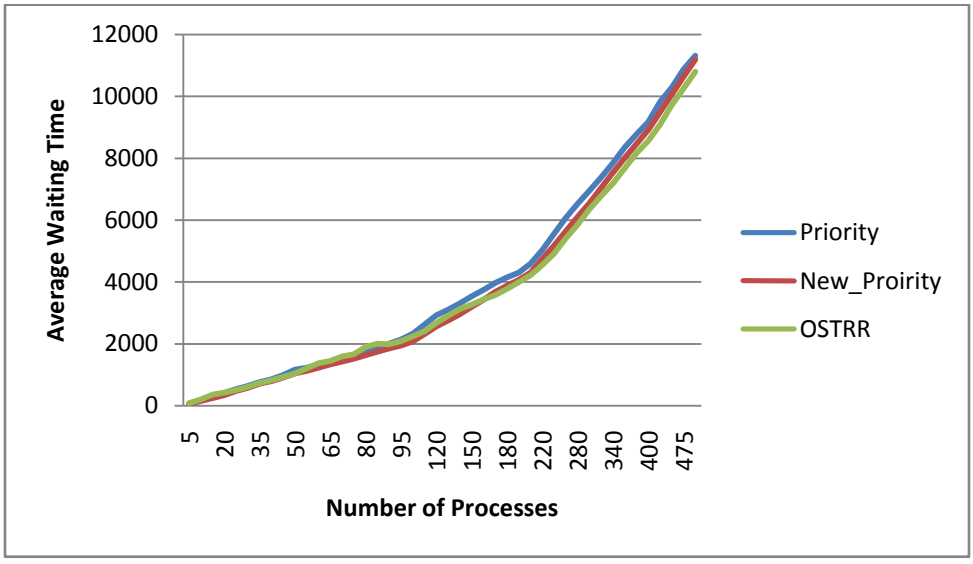


Figure4. 16: Graph of Average Waiting Time using Normal Distribution

Figure 4.16 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that OSTRR gives the least average waiting time while the proposed algorithm gives a better average waiting time than the traditional algorithm.

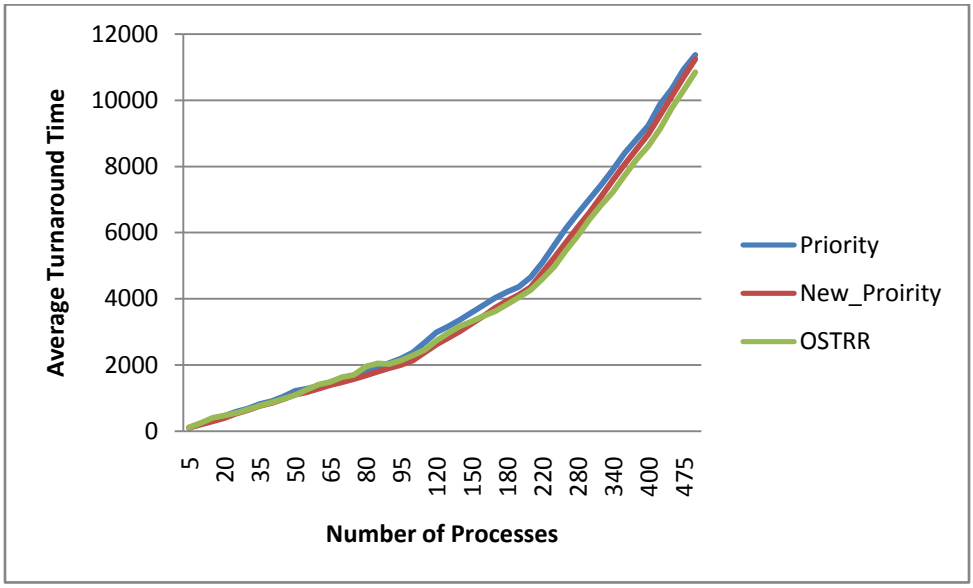
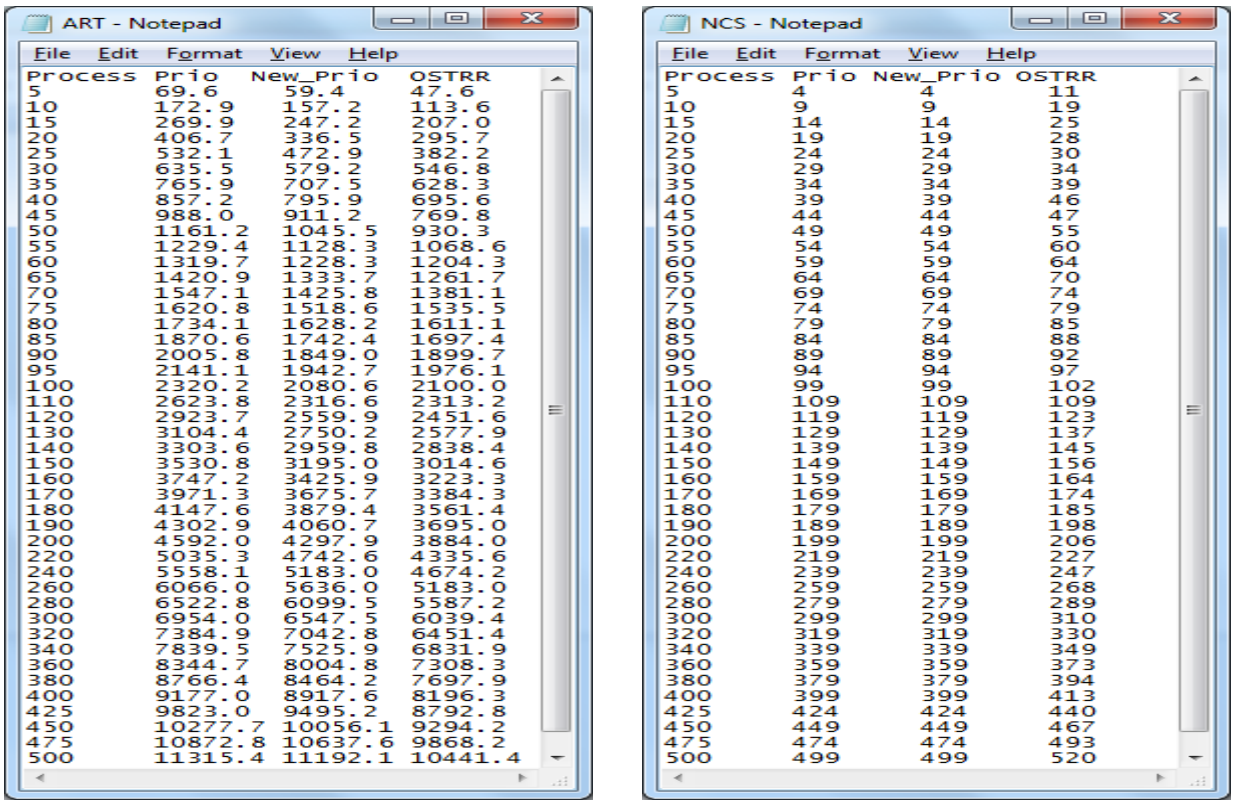


Figure4. 17: Graph of Average Turnaround Time using Normal Distribution

Figure 4.17 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that OSTRR gives the least average turnaround time while the proposed algorithm gives a better average turnaround time than the traditional algorithm.



ART - Notepad

File Edit Format View Help

Process	Prio	New_Prio	OSTRR
5	69.6	59.4	47.6
10	172.9	157.2	113.6
15	269.9	247.2	207.0
20	406.7	336.5	295.7
25	532.1	472.9	382.2
30	635.5	579.2	546.8
35	765.9	707.5	628.3
40	857.2	795.9	695.6
45	988.0	911.2	769.8
50	1161.2	1045.5	930.3
55	1229.4	1128.3	1068.6
60	1319.7	1228.3	1204.3
65	1420.9	1333.7	1261.7
70	1547.1	1425.8	1381.1
75	1620.8	1518.6	1535.5
80	1734.1	1628.2	1611.1
85	1870.6	1742.4	1697.4
90	2005.8	1849.0	1899.7
95	2141.1	1942.7	1976.1
100	2320.2	2080.6	2100.0
110	2623.8	2316.6	2313.2
120	2923.7	2559.9	2451.6
130	3104.4	2750.2	2577.9
140	3303.6	2959.8	2838.4
150	3530.8	3195.0	3014.6
160	3747.2	3425.9	3223.3
170	3971.3	3675.7	3384.3
180	4147.6	3879.4	3561.4
190	4302.9	4060.7	3695.0
200	4592.0	4297.9	3884.0
220	5035.3	4742.6	4335.6
240	5558.1	5183.0	4674.2
260	6066.0	5636.0	5183.0
280	6522.8	6099.5	5587.2
300	6954.0	6547.5	6039.4
320	7384.9	7042.8	6451.4
340	7839.5	7525.9	6831.9
360	8344.7	8004.8	7308.3
380	8766.4	8464.2	7697.9
400	9177.0	8917.6	8196.3
425	9823.0	9495.2	8792.8
450	10277.7	10056.1	9294.2
475	10872.8	10637.6	9868.2
500	11315.4	11192.1	10441.4

NCS - Notepad

File Edit Format View Help

Process	Prio	New_Prio	OSTRR
5	4	4	11
10	9	9	19
15	14	14	25
20	19	19	28
25	24	24	30
30	29	29	34
35	34	34	39
40	39	39	46
45	44	44	47
50	49	49	55
55	54	54	60
60	59	59	64
65	64	64	70
70	69	69	74
75	74	74	79
80	79	79	85
85	84	84	88
90	89	89	92
95	94	94	97
100	99	99	102
110	109	109	109
120	119	119	123
130	129	129	137
140	139	139	145
150	149	149	156
160	159	159	164
170	169	169	174
180	179	179	185
190	189	189	198
200	199	199	206
220	219	219	227
240	239	239	247
260	259	259	268
280	279	279	289
300	299	299	310
320	319	319	330
340	339	339	349
360	359	359	373
380	379	379	394
400	399	399	413
425	424	424	440
450	449	449	467
475	474	474	493
500	499	499	520

Figure4. 18: Data for Average Response Time and Number of Context Switches using Normal Distribution for Priority

Figure 4.19 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that OSTRR gives the best average response time followed by the proposed algorithm and lastly the traditional algorithm. Figure 4.20 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the proposed algorithm and the traditional algorithm give the same number of context switches while OSTRR gives a higher number of context switches.

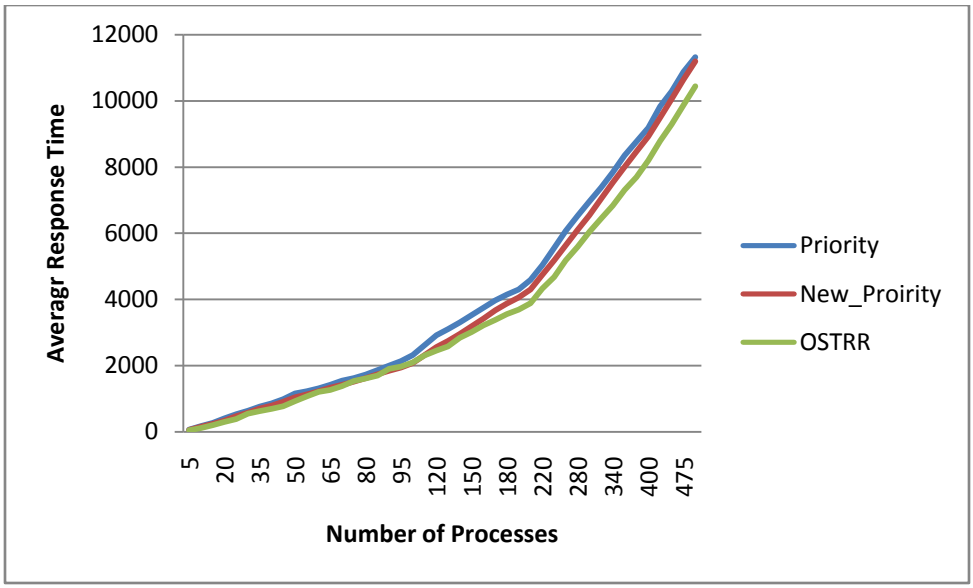


Figure4. 19: Graph of Average Response Time using Normal Distribution

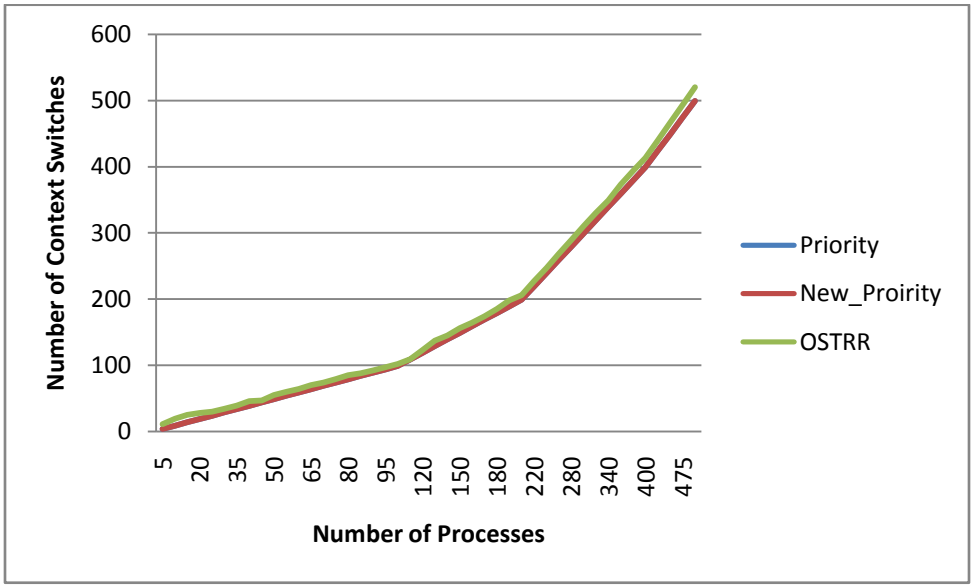
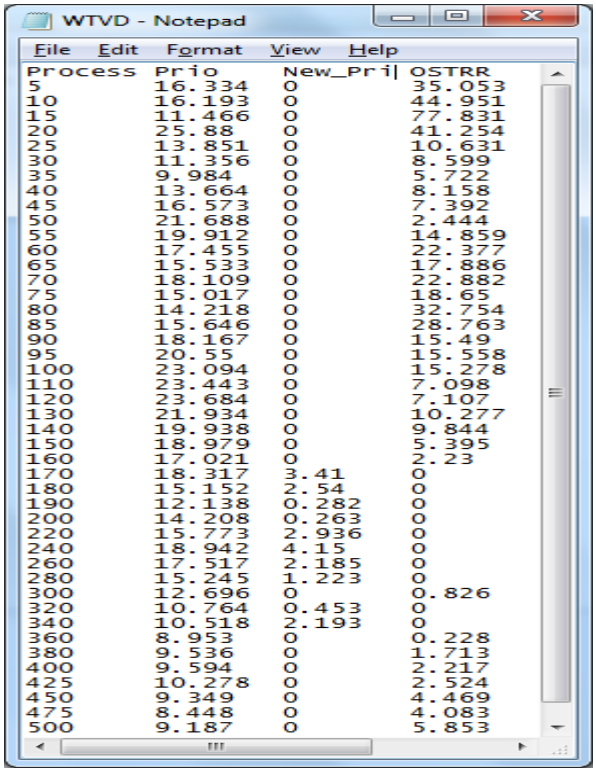


Figure4. 20: Graph of Number of Context Switches using Normal Distribution

Figure 4.22 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the proposed algorithm is the best algorithm

in terms of minimizing waiting time variance deviation followed by OSTRR and then the traditional algorithm. This shows that processes with low priorities will not starve in the proposed algorithm.



Process	Prio	New_Pri	OSTRR
5	16.334	0	35.053
10	16.193	0	44.951
15	11.466	0	77.831
20	25.88	0	41.254
25	13.851	0	10.631
30	11.356	0	8.599
35	9.984	0	5.722
40	13.664	0	8.158
45	16.573	0	7.392
50	21.688	0	2.444
55	19.912	0	14.859
60	17.455	0	22.377
65	15.533	0	17.886
70	18.109	0	22.882
75	15.017	0	18.65
80	14.218	0	32.754
85	15.646	0	28.763
90	18.167	0	15.49
95	20.55	0	15.558
100	23.094	0	15.278
110	23.443	0	7.098
120	23.684	0	7.107
130	21.934	0	10.277
140	19.938	0	9.844
150	18.979	0	5.395
160	17.021	0	2.23
170	18.317	3.41	0
180	15.152	2.54	0
190	12.138	0.282	0
200	14.208	0.263	0
220	15.773	2.936	0
240	18.942	4.15	0
260	17.517	2.185	0
280	15.245	1.223	0
300	12.696	0	0.826
320	10.764	0.453	0
340	10.518	2.193	0
360	8.953	0	0.228
380	9.536	0	1.713
400	9.594	0	2.217
425	10.278	0	2.524
450	9.349	0	4.469
475	8.448	0	4.083
500	9.187	0	5.853

Figure4. 21: Data for Waiting Time Variance Deviation using Normal Distribution for Priority

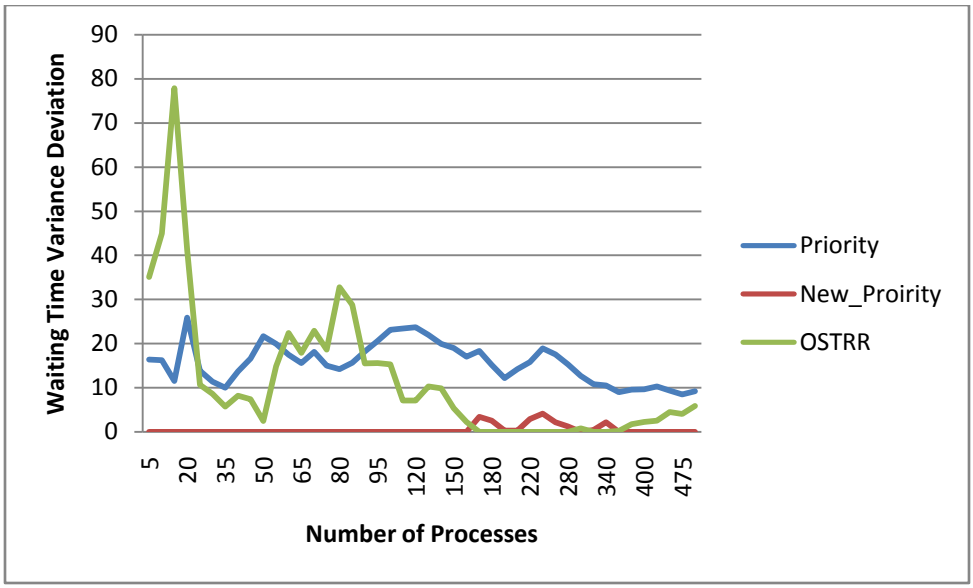


Figure4. 22: Graph of Waiting Time Variance Deviation

From the results of the two scheduling algorithms (Priority and OSTRR) compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced best Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared to the traditional Priority scheduling algorithm. But they both produced the same Number of Context Switches (NCS). The new proposed Priority algorithm produced better results compared with OSTRR with respect to Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD).

4.4.2 Results obtained using Uniform Distribution to generate burst time

The following shows the relationships between the CPU scheduling algorithms under study using 500 processes generated by uniform distribution with burst times ranging between 1 and 100*ms* and time quantum of 10*ms* used by OSTRR.

4.4.2.1 First category

AWT - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	45.6	55.4	58.4
10	81.5	114.0	153.8
15	133.5	194.9	219.9
20	187.7	270.7	279.8
25	216.7	337.4	369.0
30	221.8	380.7	416.8
35	268.6	437.0	493.3
40	300.8	495.5	527.7
45	332.5	546.3	571.5
50	363.0	591.3	644.1
55	451.1	650.6	755.1
60	500.8	720.5	778.1
65	530.0	778.6	825.7
70	567.8	834.4	887.8
75	571.2	882.7	919.9
80	597.1	934.7	1017.4
85	621.3	985.4	1072.5
90	668.0	1048.7	1137.0
95	696.1	1118.6	1267.9
100	751.6	1201.7	1329.5
110	786.5	1357.7	1400.4
120	888.7	1511.6	1521.1
130	999.4	1664.7	1687.3
140	1069.3	1807.6	1757.2
150	1129.3	1955.9	1952.4
160	1200.2	2090.6	2079.0
170	1355.9	2246.1	2288.6
180	1451.1	2398.7	2437.4
190	1526.4	2538.4	2611.9
200	1634.6	2692.0	2764.0
220	1804.3	2988.4	3109.1
240	2004.4	3298.0	3451.4
260	2167.8	3606.5	3749.1
280	2346.5	3954.5	4122.0
300	2533.9	4300.7	4449.2
320	2715.4	4648.2	4821.2
340	2812.8	4963.4	5145.9
360	2969.5	5287.5	5411.0
380	3200.9	5637.0	5818.0
400	3373.8	5972.3	6140.3
425	3544.1	6401.1	6530.5
450	3756.6	6821.7	6889.7
475	3879.2	7238.7	7253.4
500	4026.4	7659.4	7638.6

ATAT - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	45.6	55.4	40.0
10	81.5	114.0	101.2
15	133.5	194.9	173.1
20	187.7	270.7	206.7
25	216.7	337.4	265.2
30	221.8	380.7	366.7
35	268.6	437.0	426.9
40	300.8	495.5	457.3
45	332.5	546.3	534.0
50	363.0	591.3	591.7
55	451.1	650.6	709.7
60	500.8	720.5	756.3
65	530.0	778.6	794.1
70	567.8	834.4	851.6
75	571.2	882.7	883.1
80	597.1	934.7	945.0
85	621.3	985.4	991.6
90	668.0	1048.7	1057.5
95	696.1	1118.6	1112.2
100	751.6	1201.7	1249.3
110	786.5	1357.7	1369.9
120	888.7	1511.6	1480.0
130	999.4	1664.7	1625.5
140	1069.3	1807.6	1649.4
150	1129.3	1955.9	1846.2
160	1200.2	2090.6	1935.0
170	1355.9	2246.1	2143.3
180	1451.1	2398.7	2233.0
190	1526.4	2538.4	2486.7
200	1634.6	2692.0	2594.6
220	1804.3	2988.4	2857.3
240	2004.4	3298.0	3264.9
260	2167.8	3606.5	3520.5
280	2346.5	3954.5	3866.6
300	2533.9	4300.7	4222.3
320	2715.4	4648.2	4520.3
340	2812.8	4963.4	4739.5
360	2969.5	5287.5	5071.6
380	3200.9	5637.0	5363.1
400	3373.8	5972.3	5755.3
425	3544.1	6401.1	6246.6
450	3756.6	6821.7	6518.4
475	3879.2	7238.7	7068.1
500	4026.4	7659.4	7342.9

Figure4. 23: Data for Average Waiting Time and Average Turnaround Time using Uniform Distribution for Shortest Job First

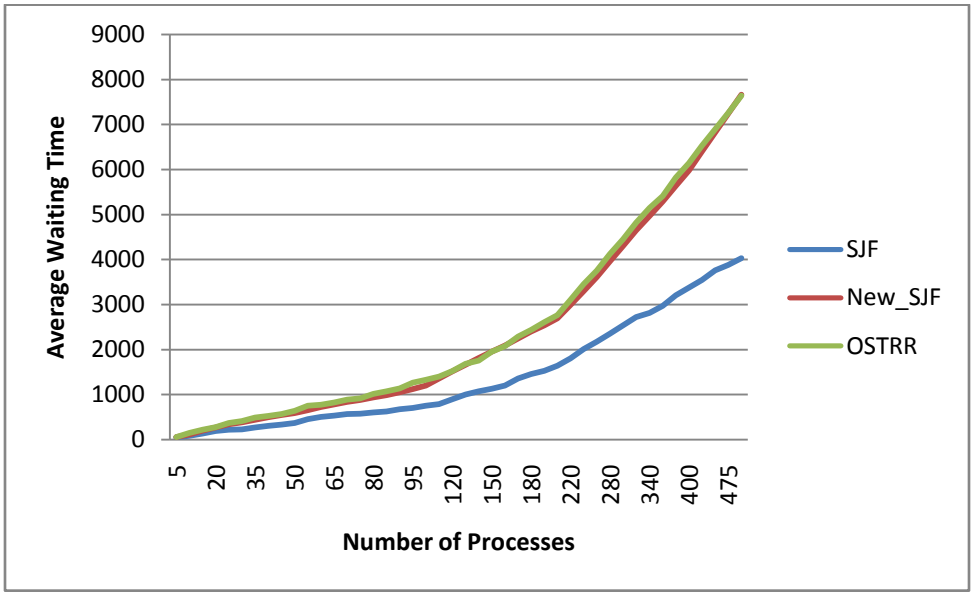


Figure4. 24: Graph of Average Waiting Time using Uniform Distribution

Figure 4.24 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing AWT.

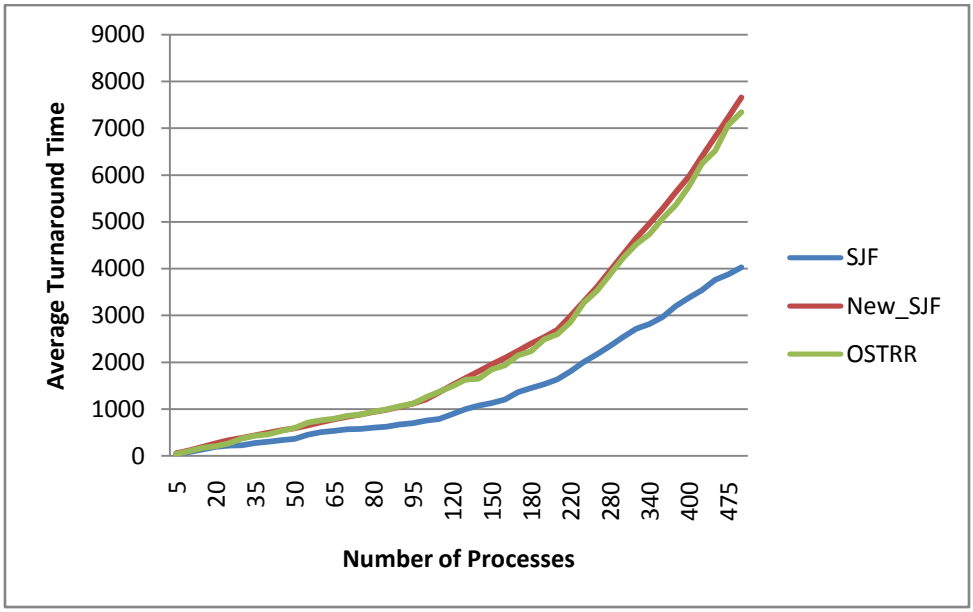
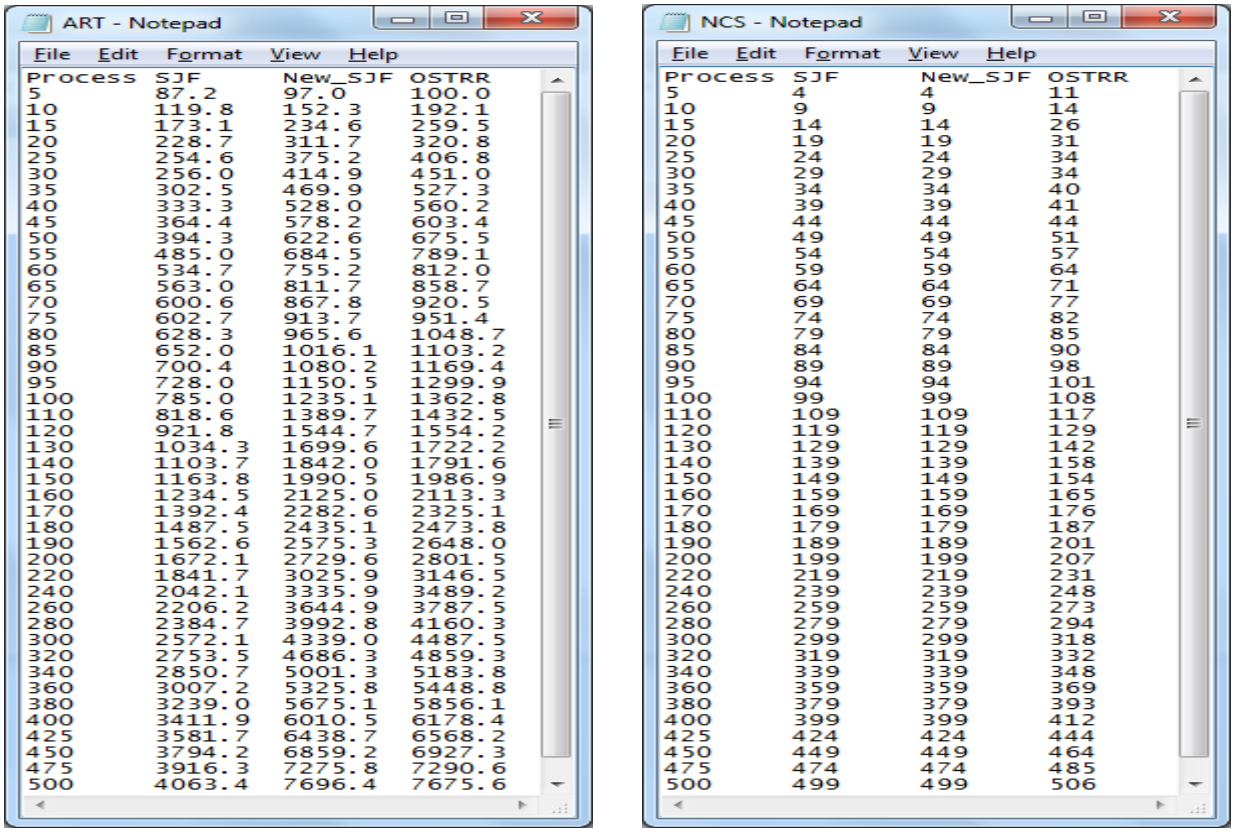


Figure4. 25: Graph of Average Turnaround Time using Uniform Distribution

Figure 4.25 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing ATAT.



ART - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	87.2	97.0	100.0
10	119.8	152.3	192.1
15	173.1	234.6	259.5
20	228.7	311.7	320.8
25	254.6	375.2	406.8
30	256.0	414.9	451.0
35	302.5	469.9	527.3
40	333.3	528.0	560.2
45	364.4	578.2	603.4
50	394.3	622.6	675.5
55	485.0	684.5	789.1
60	534.7	755.2	812.0
65	563.0	811.7	858.7
70	600.6	867.8	920.5
75	602.7	913.7	951.4
80	628.3	965.6	1048.7
85	652.0	1016.1	1103.2
90	700.4	1080.2	1169.4
95	728.0	1150.5	1299.9
100	785.0	1235.1	1362.8
110	818.6	1389.7	1432.5
120	921.8	1544.7	1554.2
130	1034.3	1699.6	1722.2
140	1103.7	1842.0	1791.6
150	1163.8	1990.5	1986.9
160	1234.5	2125.0	2113.3
170	1392.4	2282.6	2325.1
180	1487.5	2435.1	2473.8
190	1562.6	2575.3	2648.0
200	1672.1	2729.6	2801.5
220	1841.7	3025.9	3146.5
240	2042.1	3335.9	3489.2
260	2206.2	3644.9	3787.5
280	2384.7	3992.8	4160.3
300	2572.1	4339.0	4487.5
320	2753.5	4686.3	4859.3
340	2850.7	5001.3	5183.8
360	3007.2	5325.8	5448.8
380	3239.0	5675.1	5856.1
400	3411.9	6010.5	6178.4
425	3581.7	6438.7	6568.2
450	3794.2	6859.2	6927.3
475	3916.3	7275.8	7290.6
500	4063.4	7696.4	7675.6

NCS - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	4	4	11
10	9	9	14
15	14	14	26
20	19	19	31
25	24	24	34
30	29	29	34
35	34	34	40
40	39	39	41
45	44	44	44
50	49	49	51
55	54	54	57
60	59	59	64
65	64	64	71
70	69	69	77
75	74	74	82
80	79	79	85
85	84	84	90
90	89	89	98
95	94	94	101
100	99	99	108
110	109	109	117
120	119	119	129
130	129	129	142
140	139	139	158
150	149	149	154
160	159	159	165
170	169	169	176
180	179	179	187
190	189	189	201
200	199	199	207
220	219	219	231
240	239	239	248
260	259	259	273
280	279	279	294
300	299	299	318
320	319	319	332
340	339	339	348
360	359	359	369
380	379	379	393
400	399	399	412
425	424	424	444
450	449	449	464
475	474	474	485
500	499	499	506

Figure4. 26: Data for Average Response Time and Number of Context Switches using Uniform Distribution for Shortest Job First

Figure 4.27 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that the traditional algorithm SJF is better than the proposed algorithm while the proposed algorithm is better than OSTRR in some cases in terms of minimizing ART.

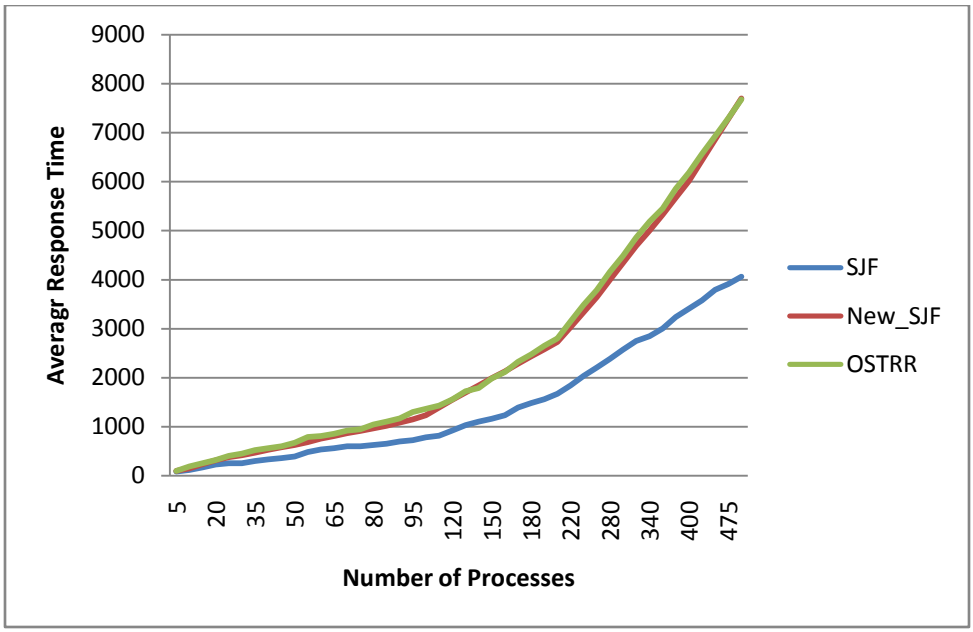


Figure4. 27: Graph of Average Response Time using Uniform Distribution

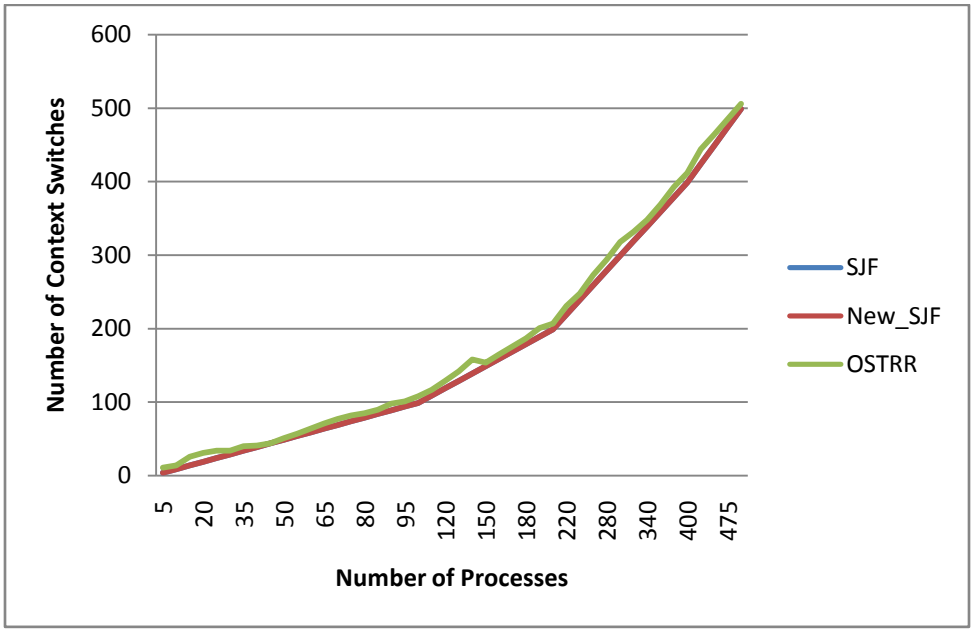
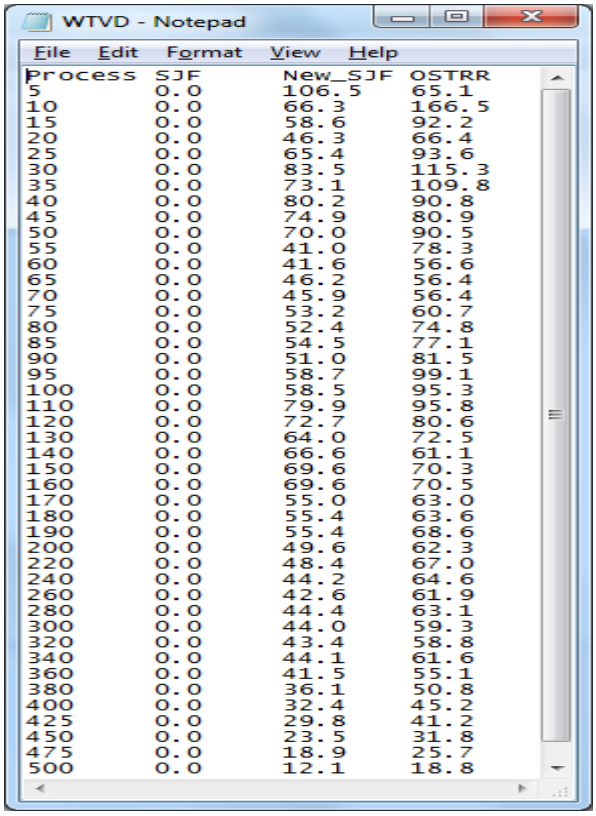


Figure4. 28: Graph of Number of Context Switches using Uniform Distribution

Figure 4.28 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the traditional algorithm and the proposed algorithm produce the same number of context switches which is better than that of OSTRR.



Process	SJF	New_SJF	OSTRR
5	0.0	106.5	65.1
10	0.0	66.3	166.5
15	0.0	58.6	92.2
20	0.0	46.3	66.4
25	0.0	65.4	93.6
30	0.0	83.5	115.3
35	0.0	73.1	109.8
40	0.0	80.2	90.8
45	0.0	74.9	80.9
50	0.0	70.0	90.5
55	0.0	41.0	78.3
60	0.0	41.6	56.6
65	0.0	46.2	56.4
70	0.0	45.9	56.4
75	0.0	53.2	60.7
80	0.0	52.4	74.8
85	0.0	54.5	77.1
90	0.0	51.0	81.5
95	0.0	58.7	99.1
100	0.0	58.5	95.3
110	0.0	79.9	95.8
120	0.0	72.7	80.6
130	0.0	64.0	72.5
140	0.0	66.6	61.1
150	0.0	69.6	70.3
160	0.0	69.6	70.5
170	0.0	55.0	63.0
180	0.0	55.4	63.6
190	0.0	55.4	68.6
200	0.0	49.6	62.3
220	0.0	48.4	67.0
240	0.0	44.2	64.6
260	0.0	42.6	61.9
280	0.0	44.4	63.1
300	0.0	44.0	59.3
320	0.0	43.4	58.8
340	0.0	44.1	61.6
360	0.0	41.5	55.1
380	0.0	36.1	50.8
400	0.0	32.4	45.2
425	0.0	29.8	41.2
450	0.0	23.5	31.8
475	0.0	18.9	25.7
500	0.0	12.1	18.8

Figure4. 29: Data for Waiting Time Variance Deviation using Uniform Distribution for Shortest Job First

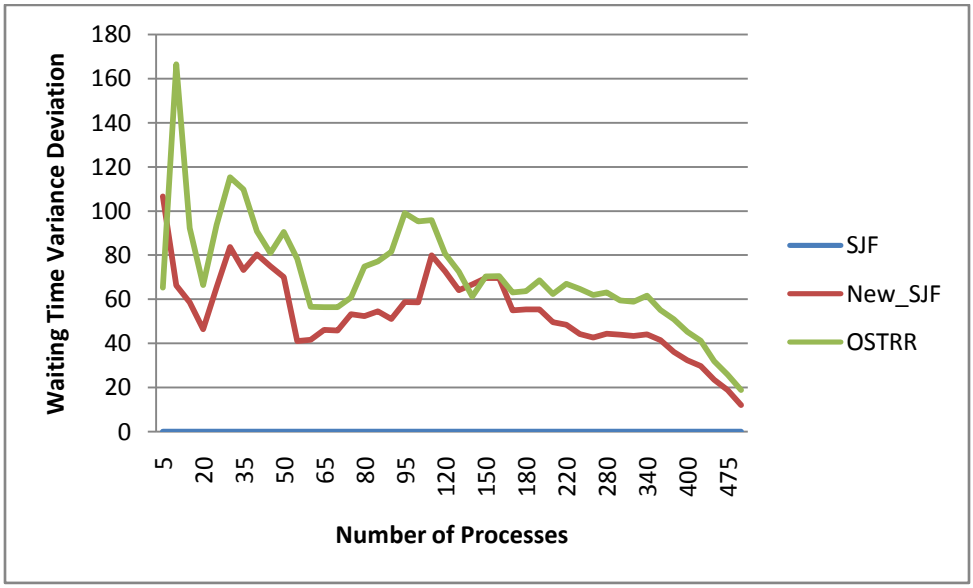


Figure4. 30: Graph of Waiting Time Variance Deviation using Uniform Distribution

Figure 4.30 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the traditional algorithm is better than the

proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing WTVD.

From the results of the two scheduling algorithms (SJF and OSTRR) compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced best Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared with the proposed new SJF. But they both produced the same Number of Context Switches (NCS). The new proposed SJF produced better results compared with OSTRR with respect to Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART), Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD).

4.4.2.2 Second category

AWT - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	81.6	53.0	58.4
10	138.2	107.5	153.8
15	236.6	194.2	219.9
20	358.0	293.4	279.8
25	424.0	365.1	369.0
30	459.9	422.3	416.8
35	539.8	483.3	493.3
40	589.0	537.6	527.7
45	665.2	600.9	571.5
50	719.2	649.8	644.1
55	832.2	712.3	755.1
60	929.5	788.8	778.1
65	988.4	863.9	825.7
70	1058.4	926.6	887.8
75	1077.4	965.5	919.9
80	1131.9	1016.9	1017.4
85	1163.8	1052.7	1072.5
90	1234.2	1111.1	1137.0
95	1276.6	1176.9	1267.9
100	1369.1	1256.9	1329.5
110	1448.2	1408.4	1400.4
120	1687.0	1559.7	1521.1
130	2039.2	1727.6	1687.3
140	2165.3	1891.5	1757.2
150	2393.6	2062.1	1952.4
160	2525.1	2207.3	2079.0
170	2917.4	2379.4	2288.6
180	3068.1	2539.5	2437.4
190	3205.4	2690.5	2611.9
200	3573.9	2861.7	2764.0
220	3874.2	3183.2	3109.1
240	4208.5	3478.3	3451.4
260	4722.8	3801.6	3749.1
280	5012.5	4130.4	4122.0
300	5340.8	4487.8	4449.2
320	5659.0	4842.5	4821.2
340	5905.3	5153.8	5145.9
360	6262.6	5485.6	5411.0
380	6644.7	5841.4	5818.0
400	6979.5	6166.3	6140.3
425	7267.3	6569.1	6530.5
450	7730.7	6987.3	6889.7
475	8029.9	7364.0	7253.4
500	8433.5	7774.0	7638.6

ATAT - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	120.8	92.2	100.0
10	175.3	147.1	192.1
15	276.3	234.7	259.5
20	397.9	335.1	320.8
25	461.9	400.9	406.8
30	494.1	456.9	451.0
35	573.1	517.3	527.3
40	622.1	572.2	560.2
45	696.5	633.1	603.4
50	749.5	681.4	675.5
55	866.7	747.3	789.1
60	962.5	822.3	812.0
65	1021.7	896.6	858.7
70	1091.7	958.9	920.5
75	1109.1	996.2	951.4
80	1163.0	1047.5	1048.7
85	1195.4	1082.5	1103.2
90	1267.4	1143.2	1169.4
95	1309.3	1209.0	1299.9
100	1403.6	1290.4	1362.8
110	1481.3	1440.6	1432.5
120	1721.0	1592.9	1554.2
130	2074.3	1762.6	1722.2
140	2200.5	1926.0	1791.6
150	2428.4	2096.8	1986.9
160	2560.1	2241.7	2113.3
170	2954.1	2416.0	2325.1
180	3104.6	2576.0	2473.8
190	3242.0	2726.7	2648.0
200	3611.5	2899.1	2801.5
220	3911.6	3220.6	3146.5
240	4246.3	3516.3	3489.2
260	4761.5	3840.1	3787.5
280	5051.1	4168.7	4160.3
300	5379.6	4526.6	4487.5
320	5697.8	4880.6	4859.3
340	5943.8	5191.9	5183.8
360	6300.7	5523.4	5448.8
380	6683.1	5879.9	5856.1
400	7018.1	6204.8	6178.4
425	7305.3	6606.8	6568.2
450	7768.7	7024.9	6927.3
475	8067.4	7401.2	7290.6
500	8470.9	7810.9	7675.6

Figure4. 31: Data for Average Waiting Time and Average Turnaround Time using Uniform Distribution for Priority

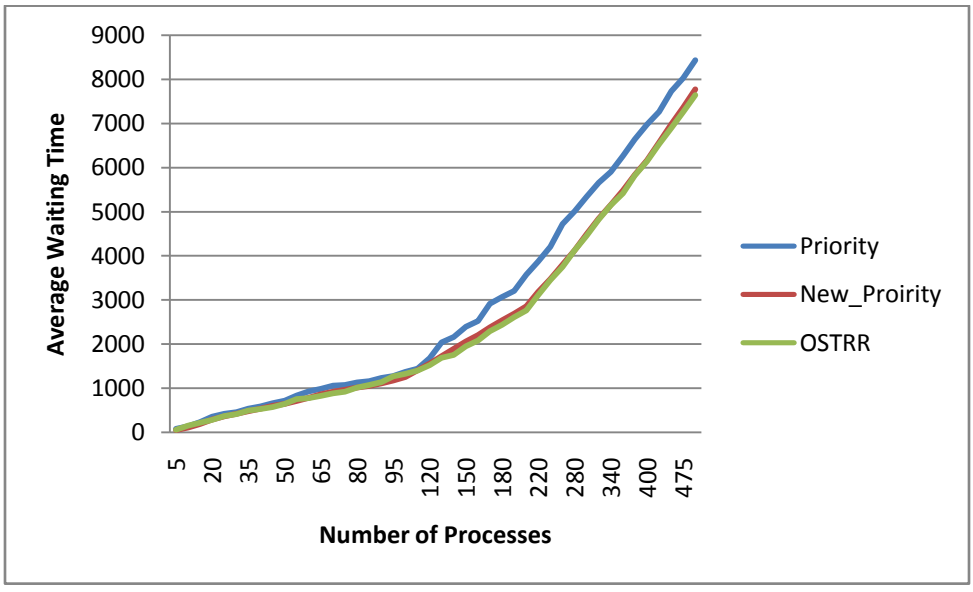


Figure4. 32: Graph of Average Waiting Time using Uniform Distribution

Figure 4.32 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that OSTRR gives the least average waiting time while the proposed algorithm gives a better average waiting time than the traditional algorithm.

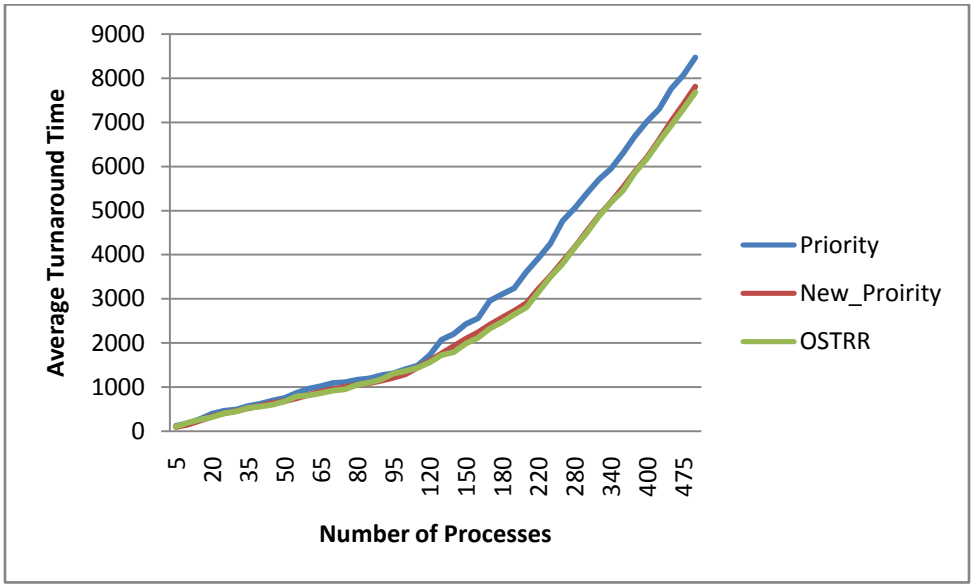
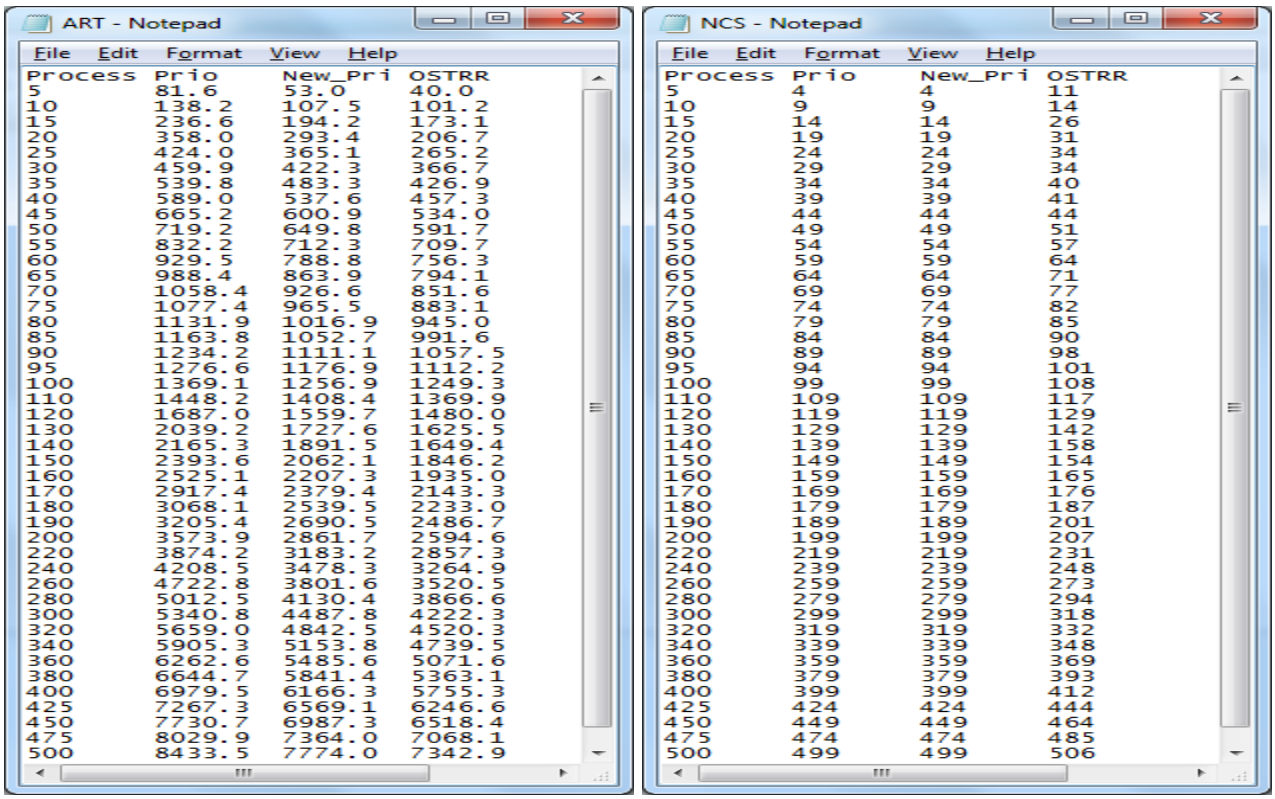


Figure4. 33: Graph of Average Turnaround Time using Uniform Distribution

Figure 4.33 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that OSTRR gives the least average turnaround time while the proposed algorithm gives a better average turnaround time than the traditional algorithm.



ART - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	81.6	53.0	40.0
10	138.2	107.5	101.2
15	236.6	194.2	173.1
20	358.0	293.4	206.7
25	424.0	365.1	265.2
30	459.9	422.3	366.7
35	539.8	483.3	426.9
40	589.0	537.6	457.3
45	665.2	600.9	534.0
50	719.2	649.8	591.7
55	832.2	712.3	709.7
60	929.5	788.8	756.3
65	988.4	863.9	794.1
70	1058.4	926.6	851.6
75	1077.4	965.5	883.1
80	1131.9	1016.9	945.0
85	1163.8	1052.7	991.6
90	1234.2	1111.1	1057.5
95	1276.6	1176.9	1112.2
100	1369.1	1256.9	1249.3
110	1448.2	1408.4	1369.9
120	1687.0	1559.7	1480.0
130	2039.2	1727.6	1625.5
140	2165.3	1891.5	1649.4
150	2393.6	2062.1	1846.2
160	2525.1	2207.3	1935.0
170	2917.4	2379.4	2143.3
180	3068.1	2539.5	2233.0
190	3205.4	2690.5	2486.7
200	3573.9	2861.7	2594.6
220	3874.2	3183.2	2857.3
240	4208.5	3478.3	3264.9
260	4722.8	3801.6	3520.5
280	5012.5	4130.4	3866.6
300	5340.8	4487.8	4222.3
320	5659.0	4842.5	4520.3
340	5905.3	5153.8	4739.5
360	6262.6	5485.6	5071.6
380	6644.7	5841.4	5363.1
400	6979.5	6166.3	5755.3
425	7267.3	6569.1	6246.6
450	7730.7	6987.3	6518.4
475	8029.9	7364.0	7068.1
500	8433.5	7774.0	7342.9

NCS - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	4	4	11
10	9	9	14
15	14	14	26
20	19	19	31
25	24	24	34
30	29	29	34
35	34	34	40
40	39	39	41
45	44	44	44
50	49	49	51
55	54	54	57
60	59	59	64
65	64	64	71
70	69	69	77
75	74	74	82
80	79	79	85
85	84	84	90
90	89	89	98
95	94	94	101
100	99	99	108
110	109	109	117
120	119	119	129
130	129	129	142
140	139	139	158
150	149	149	154
160	159	159	165
170	169	169	176
180	179	179	187
190	189	189	201
200	199	199	207
220	219	219	231
240	239	239	248
260	259	259	273
280	279	279	294
300	299	299	318
320	319	319	332
340	339	339	348
360	359	359	369
380	379	379	393
400	399	399	412
425	424	424	444
450	449	449	464
475	474	474	485
500	499	499	506

Figure4. 34: Data for Average Response Time and Number of Context Switches using Uniform Distribution for Priority

Figure 4.35 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that OSTRR gives the best average response time followed by the proposed algorithm and lastly the traditional algorithm.

Figure 4.36 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the proposed algorithm and the traditional algorithm give the same number of context switches while OSTRR gives a higher number of context switches.

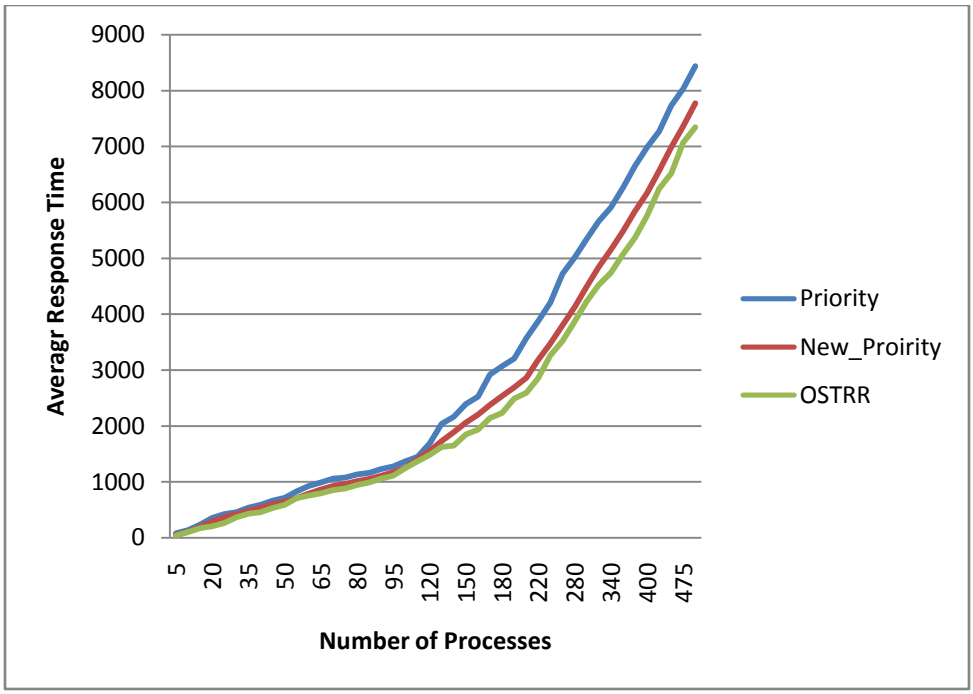


Figure4. 35: Graph of Average Response Time using Uniform Distribution

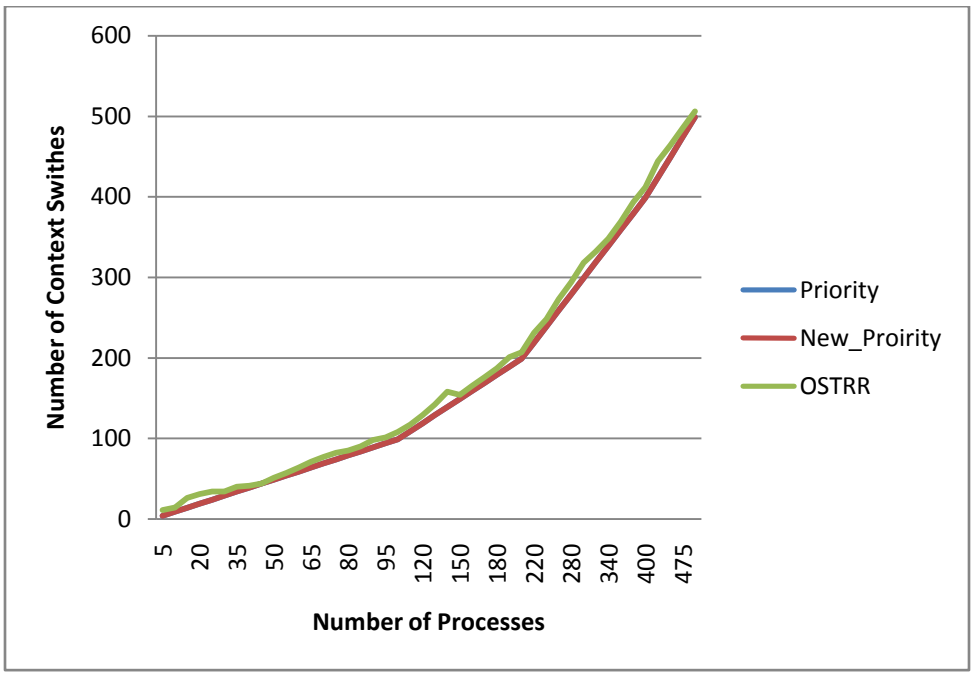
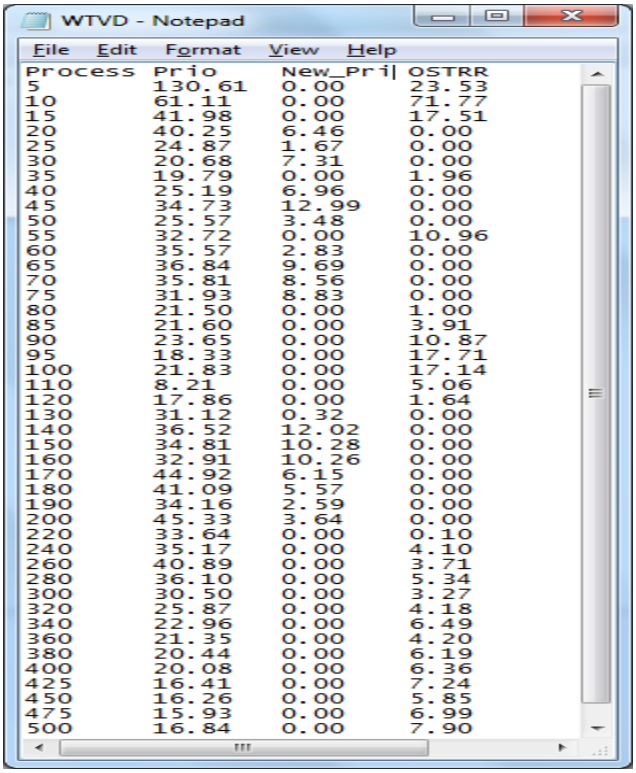


Figure4. 36: Graph of Number of Context Switches using Uniform Distribution

Figure 4.38 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the proposed algorithm is the best algorithm in terms of minimizing waiting time variance deviation followed by OSTRR and then the

traditional algorithm. This shows that processes with low priorities will not starve in the proposed algorithm.



WTVD - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	130.61	0.00	23.53
10	61.11	0.00	71.77
15	41.98	0.00	17.51
20	40.25	6.46	0.00
25	24.87	1.67	0.00
30	20.68	7.31	0.00
35	19.79	0.00	1.96
40	25.19	6.96	0.00
45	34.73	12.99	0.00
50	25.57	3.48	0.00
55	32.72	0.00	10.96
60	35.57	2.83	0.00
65	36.84	9.69	0.00
70	35.81	8.56	0.00
75	31.93	8.83	0.00
80	21.50	0.00	1.00
85	21.60	0.00	3.91
90	23.65	0.00	10.87
95	18.33	0.00	17.71
100	21.83	0.00	17.14
110	8.21	0.00	5.06
120	17.86	0.00	1.64
130	31.12	0.32	0.00
140	36.52	12.02	0.00
150	34.81	10.28	0.00
160	32.91	10.26	0.00
170	44.92	6.15	0.00
180	41.09	5.57	0.00
190	34.16	2.59	0.00
200	45.33	3.64	0.00
220	33.64	0.00	0.10
240	35.17	0.00	4.10
260	40.89	0.00	3.71
280	36.10	0.00	5.34
300	30.50	0.00	3.27
320	25.87	0.00	4.18
340	22.96	0.00	6.49
360	21.35	0.00	4.20
380	20.44	0.00	6.19
400	20.08	0.00	6.36
425	16.41	0.00	7.24
450	16.26	0.00	5.85
475	15.93	0.00	6.99
500	16.84	0.00	7.90

Figure4. 37: Data for Waiting Time Variance Deviation using Uniform Distribution for Priority

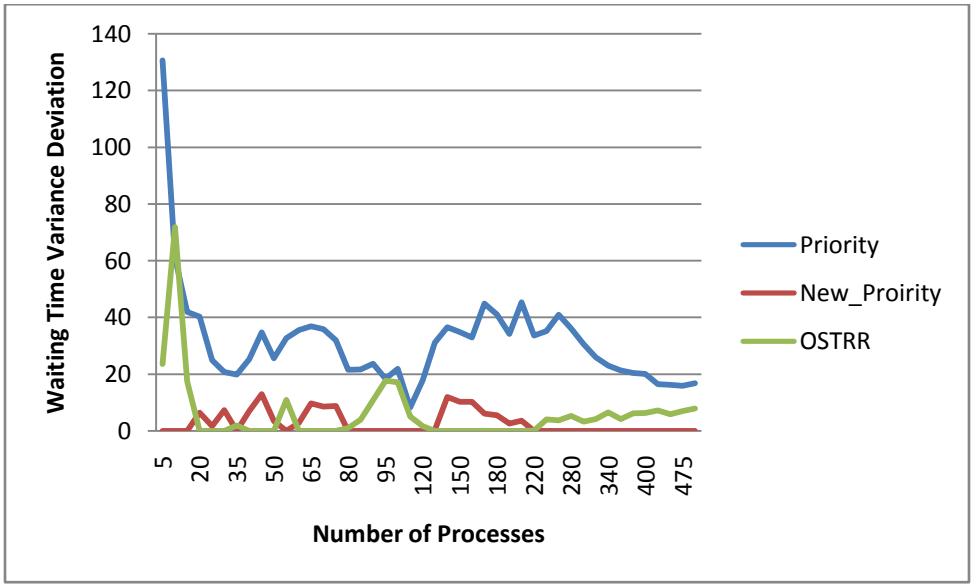


Figure4. 38: Graph of Waiting Time Variance Deviation using Uniform Distribution

From the results of the two scheduling algorithms (Priority and OSTRR) compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced best Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared to the traditional Priority scheduling algorithm. But they both produced the same Number of Context Switches (NCS). The new proposed Priority algorithm produced better results compared with OSTRR with respect to Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD).

4.4.3 Results obtained using Exponential Distribution to generate burst time

The following shows the relationships between the CPU scheduling algorithms under study using 500 processes generated by exponential distribution with a rate (λ) of 0.02 *per ms*.

4.4.3.1 First category

AWT - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	77.0	83.4	89.6
10	164.0	177.2	215.9
15	253.1	273.7	365.0
20	345.2	375.9	491.7
25	478.8	516.8	706.0
30	557.1	668.0	790.6
35	615.3	800.2	840.1
40	722.2	930.2	927.4
45	793.7	1036.8	1018.2
50	897.6	1159.9	1128.1
55	961.0	1227.7	1185.7
60	1105.7	1351.5	1323.1
65	1243.6	1472.8	1413.4
70	1376.6	1591.7	1544.8
75	1503.2	1709.7	1672.4
80	1640.4	1837.9	1771.4
85	1741.8	1943.4	1887.2
90	1837.2	2047.0	1994.8
95	1956.4	2159.1	2123.5
100	2058.5	2267.5	2293.0
110	2279.1	2493.3	2565.8
120	2505.3	2729.6	2827.9
130	2643.2	2930.0	3005.8
140	2847.1	3116.7	3206.1
150	3101.5	3355.0	3448.6
160	3342.3	3584.7	3686.8
170	3555.7	3793.4	3881.5
180	3776.4	4012.9	4114.3
190	3971.8	4220.0	4300.6
200	4166.6	4425.6	4510.3
220	4583.8	4851.5	4963.9
240	4975.4	5281.7	5411.1
260	5343.9	5702.0	5828.5
280	5714.5	6132.4	6255.3
300	6115.5	6580.8	6700.4
320	6517.1	7033.4	7166.3
340	6910.6	7497.7	7608.5
360	7330.4	7928.4	8042.1
380	7860.6	8431.7	8589.0
400	8376.5	8928.5	9077.1
425	8997.1	9544.5	9679.6
450	9594.8	10154.5	10379.2
475	10182.6	10794.1	10949.3
500	10747.7	11426.7	11534.2

ATAT - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	126.6	134.4	139.2
10	215.7	229.6	267.6
15	303.9	324.4	415.7
20	398.1	428.7	544.5
25	536.8	574.7	763.9
30	615.1	725.9	848.5
35	670.6	855.5	895.4
40	777.1	985.2	982.3
45	847.7	1090.9	1072.2
50	951.9	1214.1	1182.4
55	1012.5	1278.5	1237.2
60	1157.0	1402.2	1374.5
65	1294.8	1523.4	1464.6
70	1428.1	1642.7	1596.4
75	1555.2	1761.4	1724.4
80	1693.5	1890.4	1824.5
85	1793.8	1995.4	1939.2
90	1888.3	2097.7	2045.9
95	2008.2	2210.5	2175.4
100	2109.9	2319.4	2344.4
110	2331.0	2545.3	2617.7
120	2557.5	2781.7	2880.0
130	2694.1	2980.6	3056.7
140	2897.2	3166.5	3256.2
150	3151.7	3404.9	3498.8
160	3392.1	3634.2	3736.6
170	3604.5	3842.3	3930.3
180	3825.4	4061.7	4163.3
190	4019.9	4268.0	4348.7
200	4214.4	4473.2	4558.1
220	4632.9	4900.4	5013.0
240	5024.4	5330.7	5460.1
260	5392.7	5750.8	5877.4
280	5763.4	6181.5	6304.2
300	6164.8	6629.9	6749.7
320	6566.9	7083.1	7216.2
340	6960.8	7547.8	7658.7
360	7381.1	7978.8	8092.8
380	7911.6	8482.4	8639.9
400	8427.8	8979.5	9128.3
425	9048.6	9595.8	9731.1
450	9646.4	10205.9	10430.8
475	10234.3	10845.6	11001.1
500	10799.5	11478.4	11586.0

Figure4. 39: Data for Average Waiting Time and Average Turnaround Time using Exponential Distribution for Shortest Job First

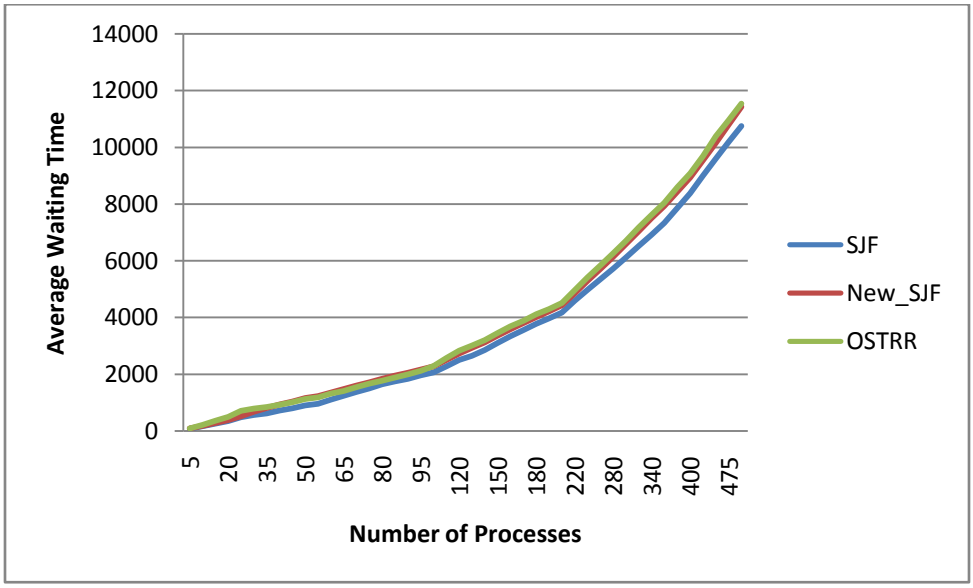


Figure4. 40: Graph of Average Waiting Time using Exponential Distribution

Figure 4.40 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing AWT.

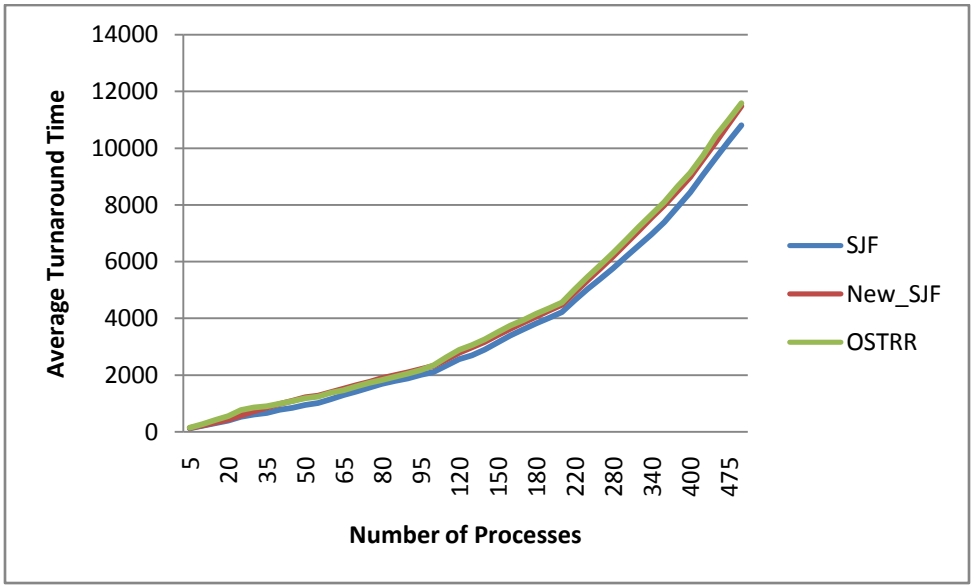


Figure4. 41: Graph of Average Turnaround Time using Exponential Distribution

Figure 4.41 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing ATAT.

ART - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	77.0	83.4	64.8
10	164.0	177.2	171.4
15	253.1	273.7	252.8
20	345.2	375.9	329.7
25	478.8	516.8	516.6
30	557.1	668.0	671.5
35	615.3	800.2	721.2
40	722.2	930.2	879.7
45	793.7	1036.8	951.2
50	897.6	1159.9	1127.2
55	961.0	1227.7	1164.8
60	1105.7	1351.5	1277.8
65	1243.6	1472.8	1403.8
70	1376.6	1591.7	1523.6
75	1503.2	1709.7	1649.7
80	1640.4	1837.9	1780.7
85	1741.8	1943.4	1895.1
90	1837.2	2047.0	1982.6
95	1956.4	2159.1	2145.1
100	2058.5	2267.5	2252.8
110	2279.1	2493.3	2453.6
120	2505.3	2729.6	2644.8
130	2643.2	2930.0	2833.4
140	2847.1	3116.7	3099.2
150	3101.5	3355.0	3334.7
160	3342.3	3584.7	3522.4
170	3555.7	3793.4	3731.3
180	3776.4	4012.9	3957.5
190	3971.8	4220.0	4151.6
200	4166.6	4425.6	4314.0
220	4583.8	4851.5	4716.6
240	4975.4	5281.7	5148.5
260	5343.9	5702.0	5559.4
280	5714.5	6132.4	5954.4
300	6115.5	6580.8	6262.2
320	6517.1	7033.4	6800.9
340	6910.6	7497.7	7367.8
360	7330.4	7928.4	7709.8
380	7860.6	8431.7	8220.2
400	8376.5	8928.5	8718.3
425	8997.1	9544.5	9247.6
450	9594.8	10154.5	9905.6
475	10182.6	10794.1	10489.2
500	10747.7	11426.7	10978.8

NCS - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	4	4	15
10	9	9	20
15	14	14	22
20	19	19	26
25	24	24	30
30	29	29	35
35	34	34	41
40	39	39	41
45	44	44	46
50	49	49	51
55	54	54	55
60	59	59	60
65	64	64	66
70	69	69	71
75	74	74	76
80	79	79	80
85	84	84	85
90	89	89	90
95	94	94	95
100	99	99	104
110	109	109	117
120	119	119	127
130	129	129	137
140	139	139	146
150	149	149	156
160	159	159	166
170	169	169	175
180	179	179	185
190	189	189	196
200	199	199	206
220	219	219	226
240	239	239	246
260	259	259	266
280	279	279	287
300	299	299	307
320	319	319	327
340	339	339	347
360	359	359	366
380	379	379	387
400	399	399	408
425	424	424	433
450	449	449	458
475	474	474	483
500	499	499	509

Figure4. 42: Data for Average Response Time and Number of Context Switches using Exponential Distribution for Shortest Job First

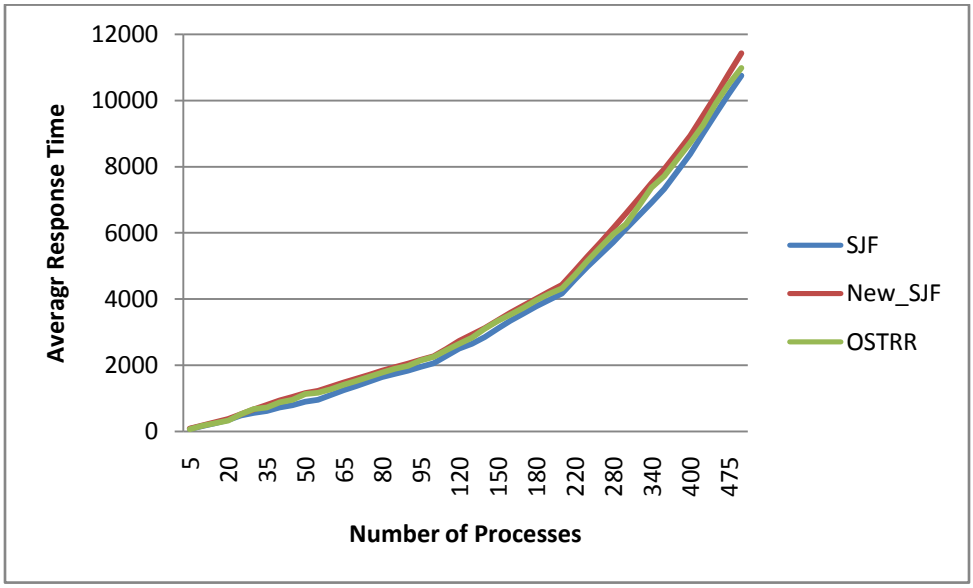


Figure4. 43: Graph of Average Response Time using Exponential Distribution

Figure 4.43 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that the traditional algorithm SJF is better than the proposed algorithm while the proposed algorithm and OSTRR produced almost the same performance in terms of minimizing ART.

Figure 4.44 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the traditional algorithm and the proposed algorithm produce the same number of context switches which is better than that of OSTRR.

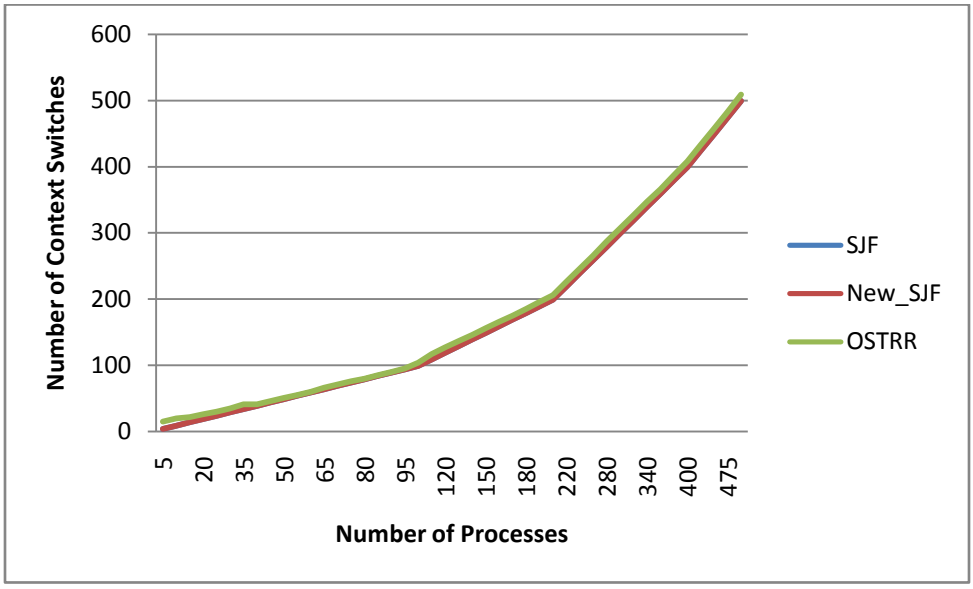
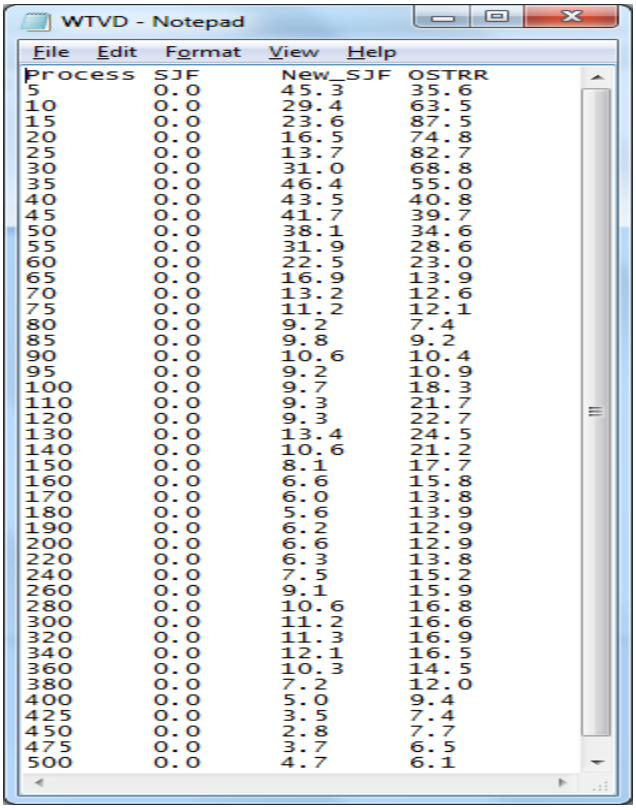


Figure4. 44: Graph of Number of Context Switches using Exponential Distribution



WTVD - Notepad

File Edit Format View Help

Process	SJF	New_SJF	OSTRR
5	0.0	45.3	35.6
10	0.0	29.4	63.5
15	0.0	23.6	87.5
20	0.0	16.5	74.8
25	0.0	13.7	82.7
30	0.0	31.0	68.8
35	0.0	46.4	55.0
40	0.0	43.5	40.8
45	0.0	41.7	39.7
50	0.0	38.1	34.6
55	0.0	31.9	28.6
60	0.0	22.5	23.0
65	0.0	16.9	13.9
70	0.0	13.2	12.6
75	0.0	11.2	12.1
80	0.0	9.2	7.4
85	0.0	9.8	9.2
90	0.0	10.6	10.4
95	0.0	9.2	10.9
100	0.0	9.7	18.3
110	0.0	9.3	21.7
120	0.0	9.3	22.7
130	0.0	13.4	24.5
140	0.0	10.6	21.2
150	0.0	8.1	17.7
160	0.0	6.6	15.8
170	0.0	6.0	13.8
180	0.0	5.6	13.9
190	0.0	6.2	12.9
200	0.0	6.6	12.9
220	0.0	6.3	13.8
240	0.0	7.5	15.2
260	0.0	9.1	15.9
280	0.0	10.6	16.8
300	0.0	11.2	16.6
320	0.0	11.3	16.9
340	0.0	12.1	16.5
360	0.0	10.3	14.5
380	0.0	7.2	12.0
400	0.0	5.0	9.4
425	0.0	3.5	7.4
450	0.0	2.8	7.7
475	0.0	3.7	6.5
500	0.0	4.7	6.1

Figure4. 45: Data for Waiting Time Variance Deviation using Exponential Distribution for Shortest Job First

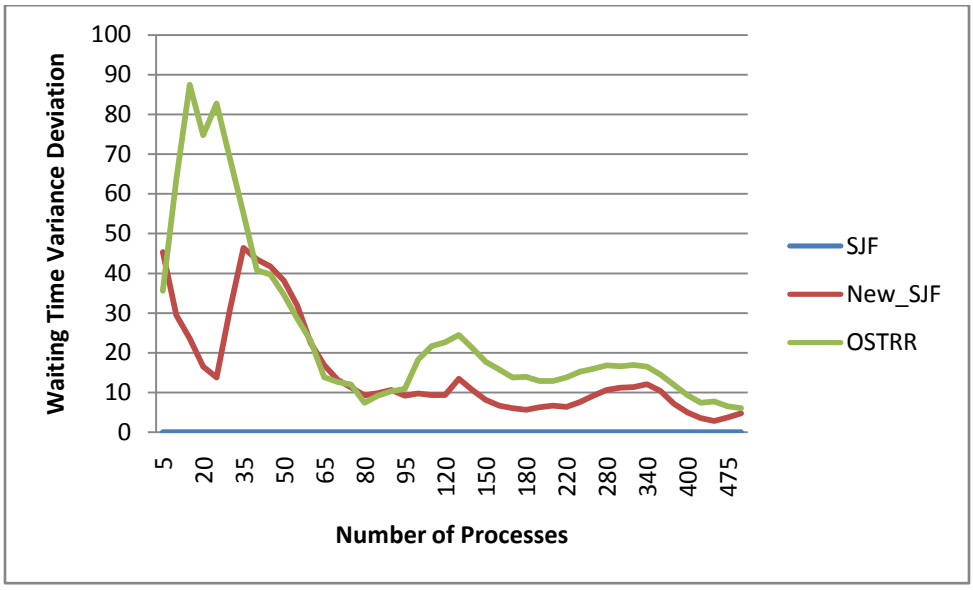


Figure4. 46: Graph of Waiting Time Variance Deviation using Exponential Distribution

Figure 4.46 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the traditional algorithm is better than the

proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing WTVD.

From the results of the two scheduling algorithms (SJF and OSTRR) compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced best Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared with the proposed new SJF. But they both produced the same Number of Context Switches (NCS). The new proposed SJF produced better results compared with OSTRR with respect to Average Waiting Time (AWT), Average Turnaround Time (ATAT), Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD).

4.4.3.2 Second category

AWT - Notepad

File Edit Format View Help

Process	Prio	New_Prio	OSTRR
5	116.0	81.4	89.6
10	224.3	186.1	215.9
15	328.1	284.9	365.0
20	486.6	400.8	491.7
25	679.3	551.1	706.0
30	863.3	706.8	790.6
35	951.8	841.8	840.1
40	1073.7	970.0	927.4
45	1190.8	1086.1	1018.2
50	1312.0	1207.0	1128.1
55	1365.6	1279.1	1185.7
60	1481.3	1400.3	1323.1
65	1597.8	1518.4	1413.4
70	1715.9	1637.6	1544.8
75	1839.6	1752.5	1672.4
80	1973.0	1877.4	1771.4
85	2070.5	1974.9	1887.2
90	2174.8	2080.5	1994.8
95	2304.3	2198.3	2123.5
100	2409.6	2300.2	2293.0
110	2672.8	2534.3	2565.8
120	2907.7	2768.5	2827.9
130	3115.4	2967.0	3005.8
140	3300.9	3157.8	3206.1
150	3537.0	3395.7	3448.6
160	3765.6	3626.9	3686.8
170	3984.5	3840.4	3881.5
180	4212.5	4056.5	4114.3
190	4428.4	4271.3	4300.6
200	4648.3	4477.0	4510.3
220	5148.1	4902.1	4963.9
240	5593.3	5336.9	5411.1
260	6051.3	5772.9	5828.5
280	6518.7	6202.8	6255.3
300	7028.4	6651.3	6700.4
320	7547.2	7104.2	7166.3
340	8077.1	7579.7	7608.5
360	8544.2	8010.3	8042.1
380	9018.6	8509.1	8589.0
400	9501.4	9004.4	9077.1
425	10106.7	9621.6	9679.6
450	10713.0	10228.9	10379.2
475	11322.9	10867.5	10949.3
500	11935.7	11500.7	11534.2

ATAT - Notepad

File Edit Format View Help

Process	Prio	New_Pri	OSTRR
5	165.6	144.4	139.2
10	276.0	244.5	267.6
15	383.3	336.1	415.7
20	539.5	455.9	544.5
25	736.5	609.3	763.9
30	921.3	764.9	848.5
35	1006.6	897.3	895.4
40	1128.6	1025.1	982.3
45	1244.5	1140.3	1072.2
50	1366.3	1261.4	1182.4
55	1416.9	1331.4	1237.2
60	1533.0	1452.3	1374.5
65	1649.0	1570.2	1464.6
70	1767.2	1689.7	1596.4
75	1892.2	1805.0	1724.4
80	2026.3	1929.9	1824.5
85	2123.0	2027.0	1939.2
90	2226.1	2132.7	2045.9
95	2356.5	2250.2	2175.4
100	2461.2	2353.5	2344.4
110	2725.4	2587.0	2617.7
120	2959.6	2820.2	2880.0
130	3167.0	3018.3	3056.7
140	3351.6	3207.8	3256.2
150	3587.7	3445.8	3498.8
160	3815.3	3676.5	3736.6
170	4033.4	3889.3	3930.3
180	4261.7	4105.5	4163.3
190	4476.6	4319.5	4348.7
200	4696.4	4525.1	4558.1
220	5197.3	4951.4	5013.0
240	5642.2	5385.9	5460.1
260	6099.9	5821.6	5877.4
280	6567.1	6251.5	6304.2
300	7077.3	6700.3	6749.7
320	7596.5	7153.8	7216.2
340	8126.8	7629.7	7658.7
360	8594.5	8060.9	8092.8
380	9069.5	8559.9	8639.9
400	9552.4	9055.5	9128.3
425	10158.0	9673.1	9731.1
450	10764.6	10280.5	10430.8
475	11374.5	10919.1	11001.1
500	11987.1	11552.4	11586.0

Figure4. 47: Data for Average Waiting Time and Average Turnaround Time using Exponential Distribution for Priority

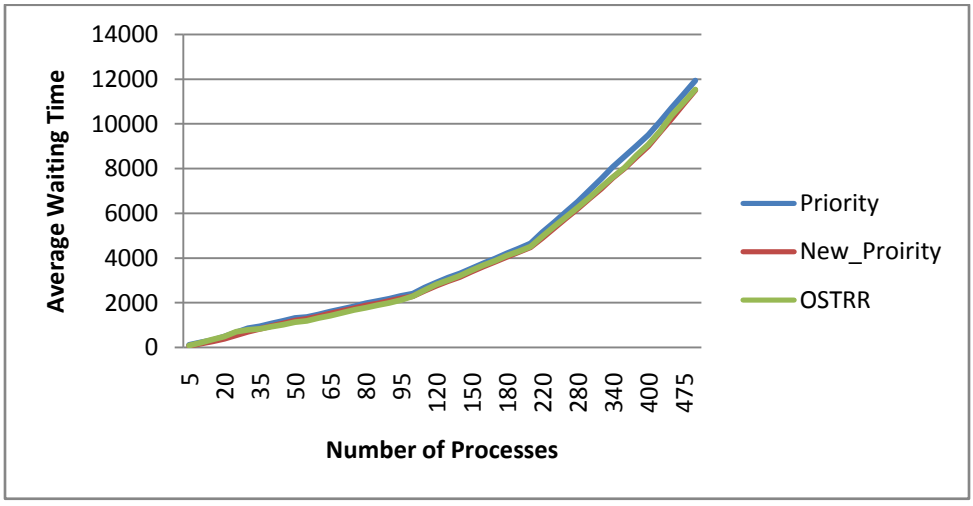


Figure4. 48: Graph of Average Waiting Time using Exponential Distribution

Figure 4.48 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that OSTRR and the proposed algorithm give the least average waiting time when compared to the traditional priority algorithm.

Figure 4.49 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that OSTRR and the proposed algorithm give the least average turnaround time when compared to the traditional priority algorithm.

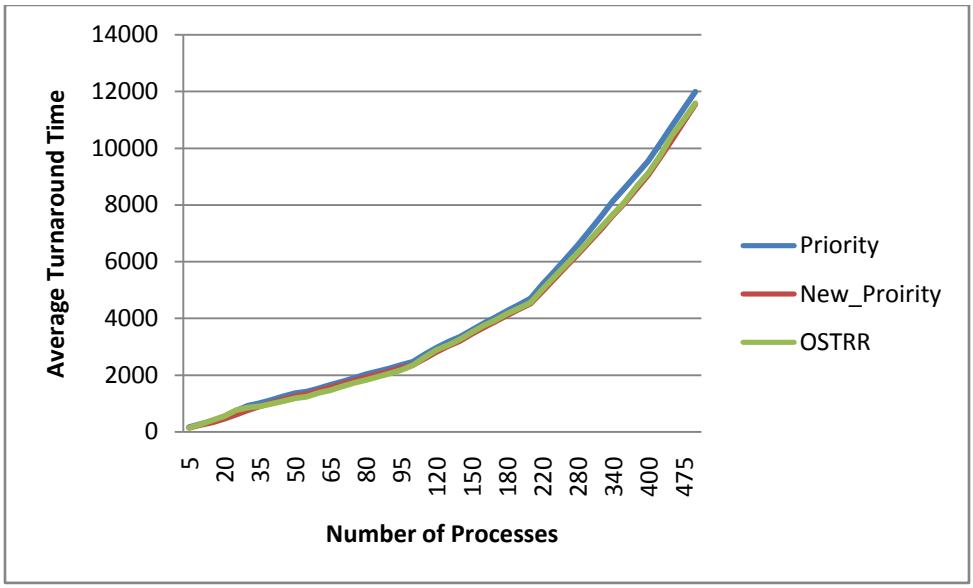
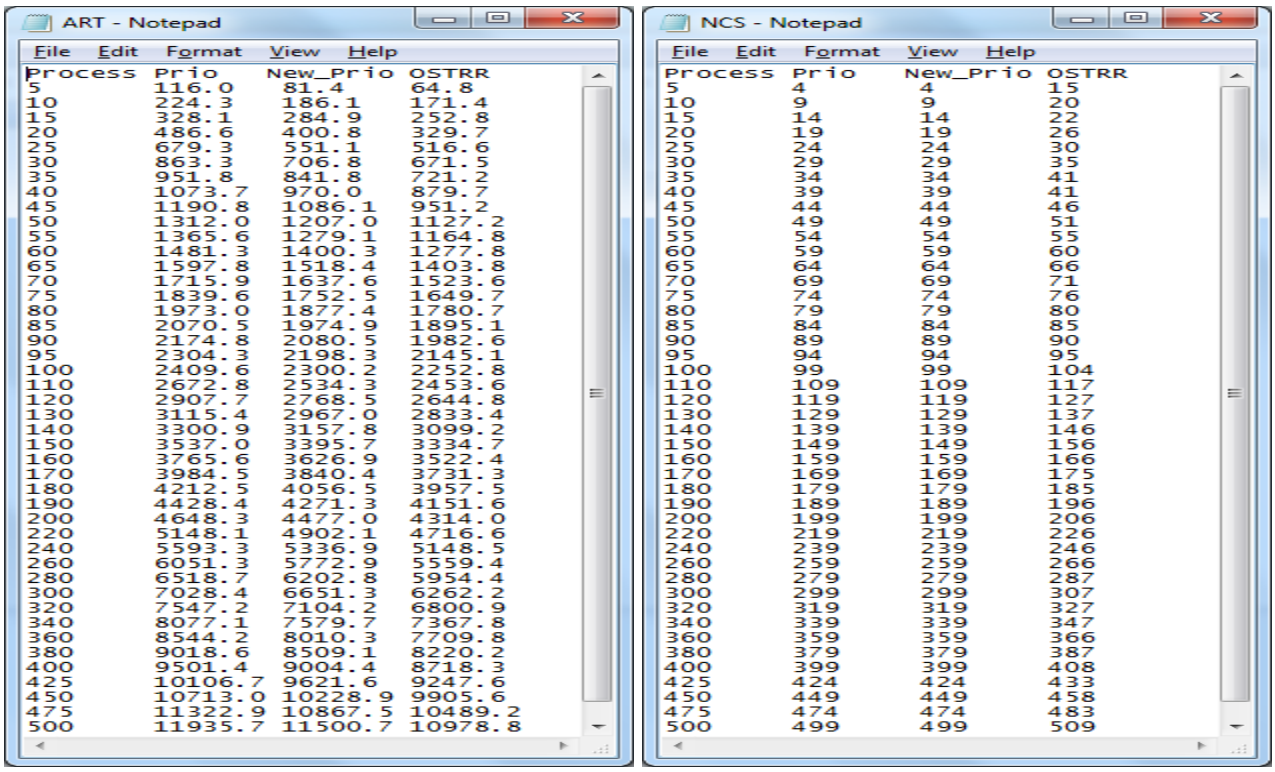


Figure4. 49: Graph of Average Turnaround Time using Exponential Distribution



ART - Notepad

File Edit Format View Help

Process	Prio	New_Prio	OSTRR
5	116.0	81.4	64.8
10	224.3	186.1	171.4
15	328.1	284.9	252.8
20	486.6	400.8	329.7
25	679.3	551.1	516.6
30	863.3	706.8	671.5
35	951.8	841.8	721.2
40	1073.7	970.0	879.7
45	1190.8	1086.1	951.2
50	1312.0	1207.0	1127.2
55	1365.6	1279.1	1164.8
60	1481.3	1400.3	1277.8
65	1597.8	1518.4	1403.8
70	1715.9	1637.6	1523.6
75	1839.6	1752.5	1649.7
80	1973.0	1877.4	1780.7
85	2070.5	1974.9	1895.1
90	2174.8	2080.5	1982.6
95	2304.3	2198.3	2145.1
100	2409.6	2300.2	2252.8
110	2672.8	2534.3	2453.6
120	2907.7	2768.5	2644.8
130	3115.4	2967.0	2833.4
140	3300.9	3157.8	3099.2
150	3537.0	3395.7	3334.7
160	3765.6	3626.9	3522.4
170	3984.5	3840.4	3731.3
180	4212.5	4056.5	3957.5
190	4428.4	4271.3	4151.6
200	4648.3	4477.0	4314.0
220	5148.1	4902.1	4716.6
240	5593.3	5336.9	5148.5
260	6051.3	5772.9	5559.4
280	6518.7	6202.8	5954.4
300	7028.4	6651.3	6262.2
320	7547.2	7104.2	6800.9
340	8077.1	7579.7	7367.8
360	8544.2	8010.3	7709.8
380	9018.6	8509.1	8220.2
400	9501.4	9004.4	8718.3
425	10106.7	9621.6	9247.6
450	10713.0	10228.9	9905.6
475	11322.9	10867.5	10489.2
500	11935.7	11500.7	10978.8

NCS - Notepad

File Edit Format View Help

Process	Prio	New_Prio	OSTRR
5	4	4	15
10	9	9	20
15	14	14	22
20	19	19	26
25	24	24	30
30	29	29	35
35	34	34	41
40	39	39	41
45	44	44	46
50	49	49	51
55	54	54	55
60	59	59	60
65	64	64	66
70	69	69	71
75	74	74	76
80	79	79	80
85	84	84	85
90	89	89	90
95	94	94	95
100	99	99	104
110	109	109	117
120	119	119	127
130	129	129	137
140	139	139	146
150	149	149	156
160	159	159	166
170	169	169	175
180	179	179	185
190	189	189	196
200	199	199	206
220	219	219	226
240	239	239	246
260	259	259	266
280	279	279	287
300	299	299	307
320	319	319	327
340	339	339	347
360	359	359	366
380	379	379	387
400	399	399	408
425	424	424	433
450	449	449	458
475	474	474	483
500	499	499	509

Figure4. 50: Data for Average Response Time and Number of Context Switches using Exponential Distribution for Priority

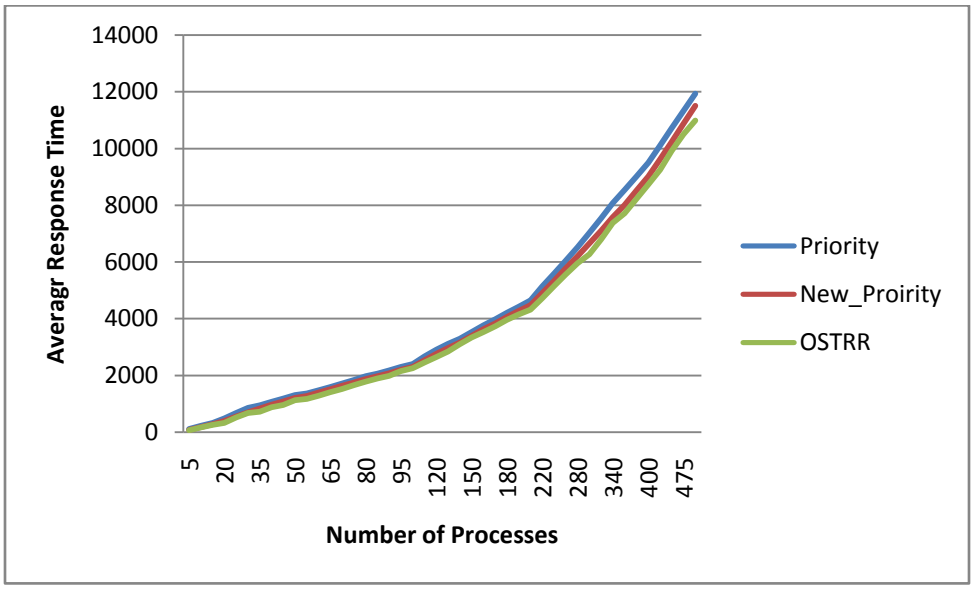


Figure4. 51: Graph of Average Response Time using Exponential Distribution

Figure 4.51 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that OSTRR gives the best average response time followed by the proposed algorithm and lastly the traditional algorithm.

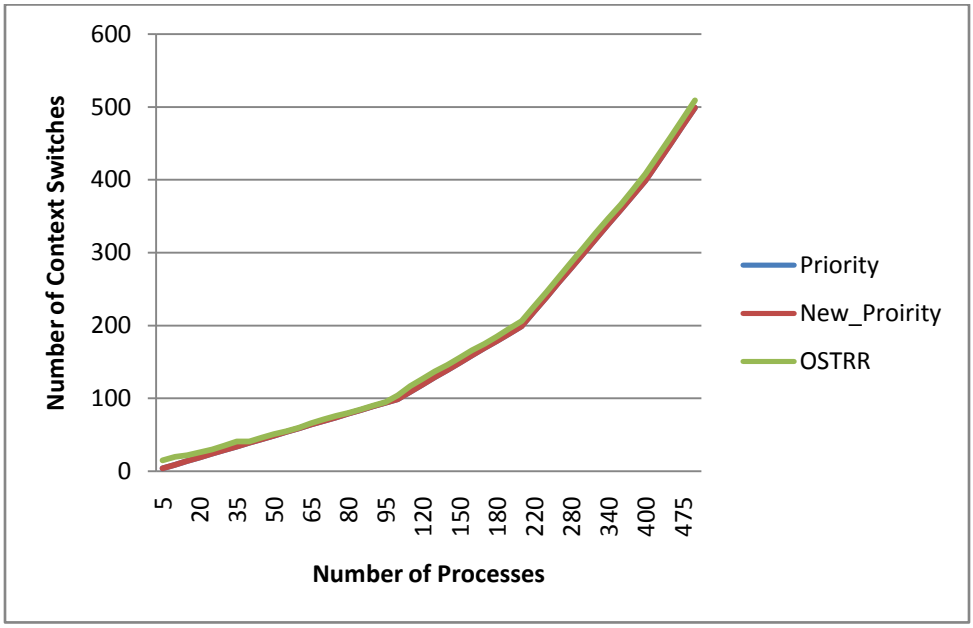
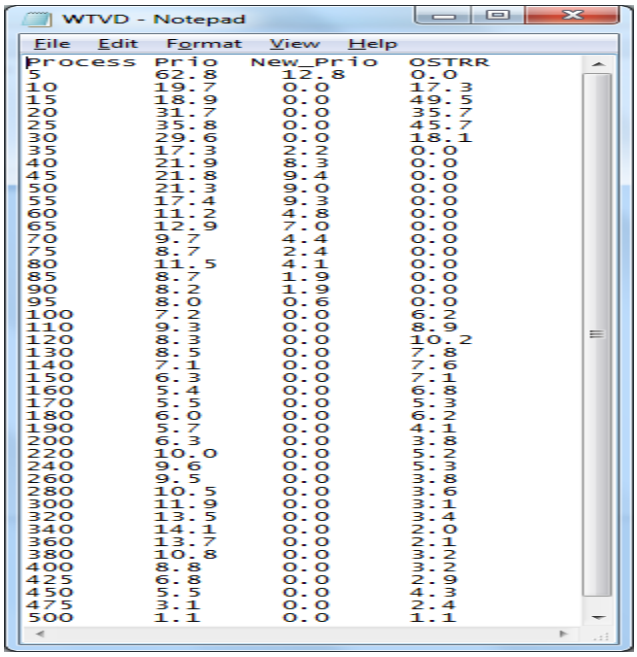


Figure4. 52: Graph of Number of Context Switches using Exponential Distribution


WTVD - Notepad

File Edit Format View Help

Process	Prio	New_Prio	OSTRR
5	62.8	12.8	0.0
10	19.7	0.0	17.3
15	18.9	0.0	49.5
20	31.7	0.0	35.7
25	35.8	0.0	45.7
30	29.6	0.0	18.1
35	17.3	2.2	0.0
40	21.9	8.3	0.0
45	21.8	9.4	0.0
50	21.3	9.0	0.0
55	17.4	9.3	0.0
60	11.2	4.8	0.0
65	12.9	7.0	0.0
70	9.7	4.4	0.0
75	8.7	2.4	0.0
80	11.5	4.1	0.0
85	8.7	1.9	0.0
90	8.2	1.9	0.0
95	8.0	0.6	0.0
100	7.2	0.0	6.2
110	9.3	0.0	8.9
120	8.3	0.0	10.2
130	8.5	0.0	7.8
140	7.1	0.0	7.6
150	6.3	0.0	7.1
160	5.4	0.0	6.8
170	5.5	0.0	5.3
180	6.0	0.0	6.2
190	5.7	0.0	4.1
200	6.3	0.0	3.8
220	10.0	0.0	5.2
240	9.6	0.0	5.3
260	9.5	0.0	3.8
280	10.5	0.0	3.6
300	11.9	0.0	3.1
320	13.5	0.0	3.4
340	14.1	0.0	2.0
360	13.7	0.0	2.1
380	10.8	0.0	3.2
400	8.8	0.0	3.2
425	6.8	0.0	2.9
450	5.5	0.0	4.3
475	3.1	0.0	2.4
500	1.1	0.0	1.1

Figure4. 53: Data for Waiting Time Variance Deviation using Exponential Distribution for Priority

Figure 4.52 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the proposed algorithm and the traditional algorithm give the same number of context switches while OSTRR gives a higher number of context switches.

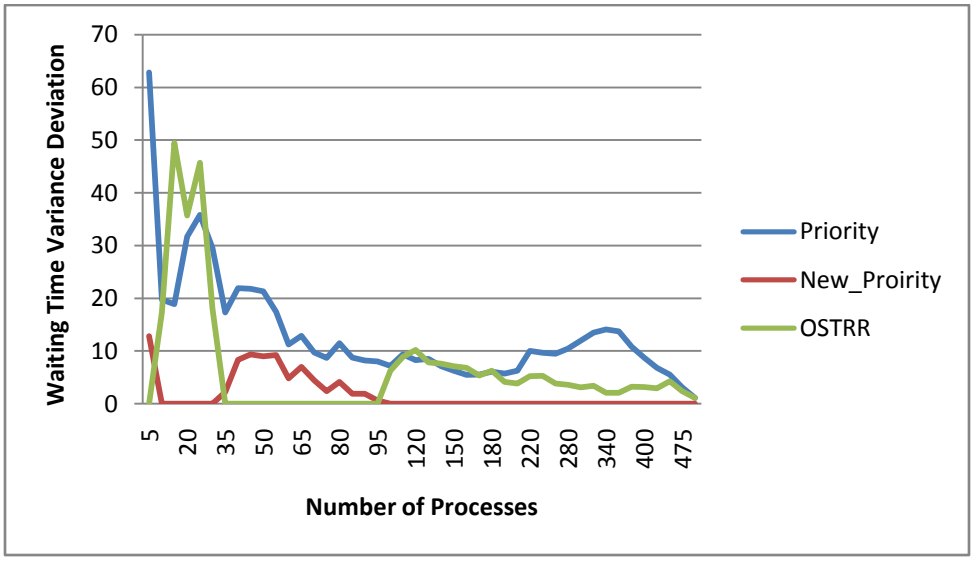


Figure4. 54: Graph of Waiting Time Variance Deviation using Exponential Distribution

Figure 4.54 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the proposed algorithm is the best algorithm in terms of minimizing waiting time variance deviation followed by OSTRR and then the traditional algorithm. This shows that processes with low priorities will not starve in the proposed algorithm.

From the results of the two scheduling algorithms (Priority and OSTRR) compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced best Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared to the traditional Priority

scheduling algorithm. But they both produced the same Number of Context Switches (NCS). The new proposed Priority algorithm produced better results compared with OSTRR with respect to Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD) also produced almost the same result in terms of Average Waiting Time (AWT), Average Turnaround Time (ATAT) and Average Response Time (ART).

Table 4.3: Performance of Statistical Distributions in Generating Burst Times

DISTRIBUTION	**CATEGORY**	**AWT** ***(ms)***	**ATAT** ***(ms)***	**ART** ***(ms)***	**NCS** ***(ms)***	**WTVD (%)**
NORMAL	FIRST(SJF)	12000	12000	12000	600	120
	SECOND(PS)	12000	12000	12000	600	90
UNIFORM	FIRST(SJF)	9000	9000	9000	600	180
	SECOND(PS)	9000	9000	9000	600	140
EXPONENTIAL	FIRST(SJF)	14000	14000	14000	600	100
	SECOND(PS)	14000	14000	14000	600	70

As shown in the table above, using the different statistical distributions in generating the burst times, it is observed that the uniform distribution produced the best result in terms of AWT, ATAT and ART in both categories. Followed by the normal distribution then lastly the exponential distribution produced the worst result. In terms of NCS, they all produced the same result in both categories. In terms of WTVD, the exponential distribution produced the best result followed normal distribution and lastly, the uniform distribution which produced the worst result.

In general, considering the first category, the traditional SJF produced better Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared with the proposed SJF. But they both produced the same Number of Context Switches (NCS). The proposed SJF produced better results compared

with OSTRR with respect to Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART), Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD). While in the second category, the proposed priority produced better Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared to the traditional Priority scheduling algorithm. But they both produced the same Number of Context Switches (NCS). The proposed Priority algorithm produced better results compared with OSTRR with respect to Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD) also produced almost the same result in terms of Average Waiting Time (AWT) Average Turnaround Time (ATAT) and Average Response Time (ART).

Table 4.4: Performance of Algorithms Based on Metrics Used in Comparing Them for First Category

Performance Metric	Traditional SJF (%)	OSTRR (%)
AWT	-47.86	-4.96
ATAT	-47.75	-1.96
ART	-47.86	-4.56
NCS	0	2.80
WTVD	-100	131.99

Table 4.4 Shows the performance of algorithms under study based on the metrics used in comparing them that is AWT, ATAT, ART, NCS and WTVD for the first category, that is the Shortest Job First Category, the comparison shows that for the AWT, ATAT and ART both the traditional Shortest Job First Algorithm and OSTRR by Saxena and Agarwal (2012) performed better than the proposed Shortest Job First Algorithm.

For the NCS, the traditional Shortest Job First Algorithm and proposed shortest job First Algorithm produced the same result which is better than the result produced by OSTRR by

Saxena and Agarwal (2012). For the WTVD, the traditional Shortest Job First Algorithm produced a better result than the Proposed Shortest Job First Algorithm, but the Proposed Shortest Job First Algorithm produced better result than OSTRR by Saxena and Agarwal (2012).

Table 4.5: Performance of Algorithms Based on Metrics Used in Comparing Them for Second Category

Performance Metric	Traditional Priority (%)	OSTRR (%)
AWT	7.36	-3.34
ATAT	7.34	-3.33
ART	7.36	-5.90
NCS	0	2.80
WTVD	16.69	12.30

Table 4.5 Shows the Performance of the Algorithms under study based on the metrics used in comparing them that is AWT, ATAT, ART, NCS and WTVD. For the second category that is the priority category, the comparison shows that the AWT, ATAT and ART, The Proposed Priority Algorithm produced better result than the Traditional Priority Algorithm but the OSTRR by Saxena and Agarwal (2012) produced better result than the Proposed Priority Algorithm.

For WTVD, the proposed priority Algorithm produced a better result than the result produced by both the Traditional Priority Algorithm and OSTRR by Saxena and Agarwal (2012).

CHAPTER FIVE: SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

The CPU is an important component of the computer system; hence it must be utilized efficiently. This can be achieved through what is called CPU scheduling which refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. The CPU scheduling is one of the most important tasks of the operating system. The basic idea is to keep the CPU busy as much as possible by executing a user process or job until it must wait for an event, and then switch to another process. The decision is made by the part of the operating system called the scheduler, using a scheduling algorithm. Some basic CPU scheduling algorithms are First Come First Served (FCFS), Shortest Job First (SJF), Priority Scheduling (PS) and Round Robin (RR). SJF and PS are both priority based CPU scheduling algorithms which assign priorities to processes based on processes burst time and priority respectively. The drawback of these algorithms is that they suffer from the problem of starvation. They are not fair as they are biased to processes of high priorities.

5.2 Conclusion

This dissertation presents an algorithm that minimized starvation in priority based CPU scheduling algorithms. This has been done by modifying the optimum priority in Saxena and Agarwal (2012) by introducing the concept of EFFICIENCY FACTOR and applying it to the priority based CPU scheduling algorithms in order to reduce average waiting time, average turnaround time, average response time and waiting time variance deviation. The priority based CPU scheduling algorithms were grouped into two groups i.e. the shortest job first and the priority groups. In each group, the proposed algorithm with the traditional algorithm in that with OSTRR were implemented in Java and their results were compared based on Average Waiting

Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART), Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD) for different categories of processes that were generated randomly using either Exponential, Uniform or Normal distributions to generate the burst time; Uniform or Exponential distributions to generate priority and Poisson distribution to generate arrival time of processes.

The simulation results show that with 500 processes generated:

1. Using Exponential distribution to generate processes burst time at a rate of 0.02, priority using Uniform distribution ranging between 1 and 50, arrival time using Poisson distribution arriving at the rate of 3 milliseconds per process and time quantum of 10*ms* used by Optimum Service Time for Round Robin algorithm.

In the SJF category, SJF and OSTRR compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced better AWT, ATAT, ART and WTVD compared with the proposed new SJF, but as number of processes exceeds 450, the proposed algorithm produced the better result. But they both produced the same NCS. The new proposed SJF produced better results compared with OSTRR with respect to NCS and WTVD. And in the PS category, Priority and OSTRR compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced better AWT, ATAT, ART and WTVD compared to the traditional Priority scheduling algorithm. But they both produced the same NCS. The new proposed Priority algorithm produced better results compared with OSTRR with respect to NCS and WTVD.

2. Using Uniform Distribution ranging between 1 and 100, priority using Exponential distribution time at a rate of 0.02, arrival time using Poisson distribution arriving at the rate of 3 milliseconds per process and time quantum of 10*ms* used by OSTRR.

In the SJF category, SJF and OSTRR compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced better AWT, ATAT, ART and WTVD compared with the proposed new SJF. But they both produced the same NCS. The new proposed SJF produced better results compared with OSTRR with respect to AWT, ATAT, ART, NCS and WTVD. And in the PS category, Priority and OSTRR compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced better AWT, ATAT, ART and WTVD compared to the traditional Priority scheduling algorithm. But they both produced the same NCS. The new proposed Priority algorithm produced better results compared with OSTRR with respect to NCS and WTVD.

3. Using Normal distribution to generate processes burst time with a mean of 50.5*ms*, standard deviation of 28.6*ms*, priority using Exponential distribution to generate processes burst time at a rate of 0.02, arrival time using Poisson distribution arriving at the rate of 3 milliseconds per process and time quantum of 10*ms* used by OSTRR.

In the SJF category, SJF and OSTRR compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced better AWT, ATAT, ART and WTVD compared with the proposed new SJF. But they both produced the same NCS. The new proposed SJF produced better results compared with OSTRR with respect to AWT, ATAT, NCS and WTVD. And in the PS category, Priority and OSTRR compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced better AWT, ATAT, ART and

WTVD compared to the traditional Priority scheduling algorithm. But they both produced the same NCS. The new proposed Priority algorithm produced better results compared with OSTRR with respect to NCS and WTVD also produced almost the same result in terms of AWT, ARTh and ATAT.

In general, in all the categories, the traditional SJF produced better AWT, ATAT, ART and WTVD compared with the proposed new SJF. But they both produced the same NCS. The new proposed SJF produced better results compared with OSTRR with respect to AWT, ATAT, NCS and WTVD. But, in the priority category, the proposed priority algorithm produced better results than the traditional priority algorithm. They both produced the same number of context switches. The new proposed priority algorithm produced a better result than OSTRR in terms of NCS and WTVD. It also produced almost the same result with OSTRR in terms of AWT, ATAT and ART.

5.3 Recommendation

The recommendation in this dissertation is to implement the Proposed Priority Algorithm instead of the Traditional Priority Algorithm, so as to minimize starvation of processes. As a future work, a more efficient algorithm should be developed to minimize starvation in the Shortest Job First (SJF) Algorithm.

REFERENCES

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

ABDULLAHI, I. (2012). *PROCESS SCHEDULING IN LONGEST JOB FIRST (LJF) ALGORITHM: A Proposed framework for Starvation Problem.* A.B.U Zaria: M.Sc Thesis.

Abdulrazaq, A., Saleh, E. A., and Junaidu, B. S. (2014). A New Improved Round Robin (NIRR) CPU Scheduling Algorithm. *International Journal of Computer Applications, 90* (4), 27-33.

Behera, H. S., Rakesh, M., Sabyasachi, S., and Sourav, K. B. (2011). COMPARATIVE PERFORMANCE ANALYSIS OF MULTI-DYNAMIC TIME QUANTUM ROUND ROBIN (MDTQRR) ALGORITHM WITH ARRIVAL TIME. *Indian Journal of Computer Science and Engineering (IJCSE)* , 262-271.

Behera, H.S, Mohanty, R and Debashree, N. (2010). A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis. *International Journal of Computer Applications (0975 – 8887), 5* (5), 10-15.

Brinch, H. P. (1977). *The Architecture of Concurrent Programs.* Englewood Cli®s, NJ: Prentice Hall.

Galvin, P. B., Abrahm, S., and Gagne, G. (2005). *Ch.5 Operating System Concepts (7th ed.). (B.Zobrist, K. Santor, & M. Lesure, Eds.).* NJ, USA: John Wiley and Sons Inc.

http://cs.engr.uky.edu/~lewis/essays/algorithms/complexity/complex.html . (n.d.). Retrieved March 7, 2015, from Complexity and Analysis of Algorithms: http://cs.engr.uky.edu/~lewis/essays/algorithms/complexity/complex.html

http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node122.html. (n.d.). Retrieved May 25, 2014, from http://siber.cankaya.edu.tr.

http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node123.html. (n.d.). Retrieved May 25, 2014, from http://siber.cankaya.edu.tr.

http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node124.html. (n.d.). Retrieved May 25, 2014, from http://siber.cankaya.edu.tr.

http://www.merriam-webster.com/dictionary/variance. (n.d.). Retrieved June 28, 2014, from http://www.merriam-webster.com.

https://www.it.uu.se/edu/course/.../scheduling_algorithms/handout.pdf. (n.d.). Retrieved May 25, 2014, from https://www.it.uu.se/edu/course/.../scheduling_algorithms/handout.pdf.

Jerry, B., John, S. C., Barry, L. N., and David, K. (2005). *Discrete-Event System Simulation* (Fourth ed.). Pearson Education International.Upper Saddle River, NJ 07458, 169-179, 272-296

LEONARD, K. (1970). Continuum of Time Sharing Scheduling Algorithms. *Spring Joint Computer Conference* , 453-458.

Letkowski, J. (n.d.). *Applications of the Poisson probability distribution.* Retrieved May 7, 2015, from http://www.aabri.com/SA12Manuscripts/SA12083.pdf.

MEHDI, N., MEHDI, S., ADEL, N., AND ALI, A. (2012). THE NEW METHOD OF ADAPTIVE CPU SCHEDULING USING FONSECA AND FLEMING'S GENETIC ALGORITHM. *Journal of Theoretical and Applied Information Technology* , 1-16.

Nong, Y., Xueping, L., Toni, F., and Xiaoyun, X. (2007). Job scheduling methods for reducing waiting time variance. *Computers and Operations Research* , 3069 - 3083.

Normal distribution. (n.d.). Retrieved May 7, 2015, from https://math.berkeley.edu/~scanlon/m16bs04/ln/16b2lec31.pdf.

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

Rakesh, M, Manas, D, Lakshmi, M. P and Sudhashree. (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), 23-27.

Saxena, H. F., and Agarwal, P. S. (2012). Design and Performance Evaluation of Optimum Service Time Concept for Round Robin Algorithm. *International Journal of Machine Learning and Computing, 2* (2), 113-117.

Silberschatz, P. B. Galvin and G. Gagne. (2006). *Operating System Concepts.* (7th, Ed.) John Wiley and Sons Inc. 111 River Street, Hoboken NJ, 07030, 153-168

Soraj, H and Roy, K.C. (n.d.). *Adaptive Round Robin scheduling using shortest burst approach, based on smart time slice*. Retrieved December 10th, 2014, from International Journal of Data Engineering: www.cscjournals.org/csc/manuscript/Journals/IJDE/.../IJDE-57.pdf

Stallings, W. (2001). *Operating Systems, Internals and design Principles.* (4th, Ed.) CA, USA: Prentice Hall.

Suri, P.K and Sumit, M. (2012). Design of Stochastic Simulator for Analyzing the Impact of Scalability on CPU Scheduling Algorithms. *International Journal of Computer Applications (0975 – 8887), 49* (17), 4-9.

The Exponential Distribution. (n.d.). Retrieved May 7, 2015, from http://reliawiki.org/index.php/The_Exponential_Distribution.

Uniform Distribution. (n.d.). Retrieved May 7, 2015, from http://www.itl.nist.gov/div898/handbook/eda/section3/eda3662.htm.

Zindros, D. (n.d.). *http://discrete.gr/complexity/* . Retrieved February 22, 2015, from A Gentle Introduction to Algorithm Complexity Analysis: http://discrete.gr/complexity/

APPENDIX

// Proposed Priority Algorithm

```
   public double [][] New_Priority1(double []burst_value,int []arrival_value,double
[]prio_value,intnum){

double [][] new_priority=new double[7][num+1];

doubleCont_Switch;

doubleAv_WT,Av_TAT,Av_RES,Vari;

   Boolean state = false;

New_Algorithmsop[] = new New_Algorithms[num+1];       // creatng array of objects

for(int i=0;i<=num;i++)

op[i]= new New_Algorithms();

System.out.println("New_Priority Starts");

double index [] = new double[num+1];

double waiting [] = new double[num+1];

boolean constant=false;

if (step==1){

constant=true;

      }

int value=step;

for(int l=start;l<=num;l=l+value){

if(constant==true){

value=1;

          }

else{

if(l==2){

value=2;

          }

if(l==5){
```

```
value=5;
        }
if(l==100){
value=10;
        }
if(l==200){
value=20;
        }
if(l==400){
value=25;
        }
if(l==500){
value=50;
        }
if(l==1000){
value=100;
        }
if(l==2000){
value=200;
        }
if(l==4000){
value=200;
        }
if(l==10000){
value=250;
        }
if(l>10000){
```

```
value=500;
        }
        }
int sum=0;
int t=0;
for(int m=1;m<=l;m++)
{
op[m].pr=m;
op[m].bt=burst_value[m];
op[m].at=arrival_value[m];
op[m].pri=prio_value[m];
sum+=op[m].bt;
op[m].copy_of_bt=op[m].bt;
}
double response = 0, temp=0, context=-1,index_cs=0, boundary=0,val=0;
intind=0;
while(t<sum)
{
for(int k=1;k<=l;k++)
    {
for(int i=1;i<l;i++)
    {

if((op[i+1].pri<op[i].pri)&&(op[i+1].bt>0)&&(op[i].bt>0)&&(op[i].at<=boundary)&&(op[i+1].
at<=boundary))
        {
temp=op[i+1].pri;
op[i+1].pri=op[i].pri;
```

```
op[i].pri=temp;
temp=op[i+1].at;
op[i+1].at=op[i].at;
op[i].at=temp;
temp=op[i+1].bt;
op[i+1].bt=op[i].bt;
op[i].bt=temp;
temp=op[i+1].pr;
op[i+1].pr=op[i].pr;
op[i].pr=temp;
temp = op[k+1].copy_of_bt;
op[k+1].copy_of_bt = op[k].copy_of_bt;
op[k].copy_of_bt = temp;
temp=op[i+1].bw;
op[i+1].bw=op[i].bw;
op[i].bw=temp;
temp=op[i+1].ww;
op[i+1].ww=op[i].ww;
op[i].ww=temp;
temp=op[i+1].pw;
op[i+1].pw=op[i].pw;
op[i].pw=temp;
temp=op[i+1].eff;
op[i+1].eff=op[i].eff;
op[i].eff=temp;
        }
      }
```

```
   }
doublepri_pos=0,temp_pri=0;
for(int i=1;i<=l;i++)
      {
if((op[i].bt>0)&&(op[i].at<=boundary))
         {
if((op[i].pri==temp_pri)){
pri_pos=pri_pos+0;
op[i].pw=pri_pos;
temp_pri=op[i].pri;
            }
else{
pri_pos=pri_pos+1;
op[i].pw=pri_pos;
temp_pri=op[i].pri;
             }
         }
      }
for(int k=1;k<=l;k++)
      {
for(int i=1;i<l;i++)
      {

if((op[i+1].bt<op[i].bt)&&(op[i+1].bt>0)&&(op[i].bt>0)&&(op[i].at<=boundary)&&(op[i+1].at
<=boundary))
         {
temp=op[i+1].pri;
op[i+1].pri=op[i].pri;
```

```
op[i].pri=temp;
temp=op[i+1].at;
op[i+1].at=op[i].at;
op[i].at=temp;
temp=op[i+1].bt;
op[i+1].bt=op[i].bt;
op[i].bt=temp;
temp=op[i+1].pr;
op[i+1].pr=op[i].pr;
op[i].pr=temp;
temp = op[k+1].copy_of_bt;
op[k+1].copy_of_bt = op[k].copy_of_bt;
op[k].copy_of_bt = temp;
temp=op[i+1].bw;
op[i+1].bw=op[i].bw;
op[i].bw=temp;
temp=op[i+1].ww;
op[i+1].ww=op[i].ww;
op[i].ww=temp;
temp=op[i+1].pw;
op[i+1].pw=op[i].pw;
op[i].pw=temp;
temp=op[i+1].eff;
op[i+1].eff=op[i].eff;
op[i].eff=temp;
        }
      }
```

```
   }
doublebt_pos=0,temp_bt=0;
for(int i=1;i<=l;i++)
      {
if((op[i].bt>0)&&(op[i].at<=boundary))
         {
if((op[i].bt!=temp_bt)){
bt_pos=bt_pos+1;
op[i].bw=bt_pos;
temp_bt=op[i].bt;
            }
else{
op[i].bw=bt_pos;
temp_bt=op[i].bt;
            }
        }
      }
for(int k=1;k<=l;k++)
      {
for(int i=1;i<l;i++)
      {

if((op[i+1].at<op[i].at)&&(op[i+1].bt>0)&&(op[i].bt>0)&&(op[i].at<=boundary)&&(op[i+1].at
<=boundary))
         {
temp=op[i+1].pri;
op[i+1].pri=op[i].pri;
op[i].pri=temp;
```

```
temp=op[i+1].at;
op[i+1].at=op[i].at;
op[i].at=temp;
temp=op[i+1].bt;
op[i+1].bt=op[i].bt;
op[i].bt=temp;
temp=op[i+1].pr;
op[i+1].pr=op[i].pr;
op[i].pr=temp;
temp = op[k+1].copy_of_bt;
op[k+1].copy_of_bt = op[k].copy_of_bt;
op[k].copy_of_bt = temp;
temp=op[i+1].bw;
op[i+1].bw=op[i].bw;
op[i].bw=temp;
temp=op[i+1].ww;
op[i+1].ww=op[i].ww;
op[i].ww=temp;
temp=op[i+1].pw;
op[i+1].pw=op[i].pw;
op[i].pw=temp;
temp=op[i+1].eff;
op[i+1].eff=op[i].eff;
op[i].eff=temp;
          }
       }
   }
```

```
doubleat_pos=0,temp_at=-1;
for(int i=1;i<=l;i++)
	{
if((op[i].bt>0)&&(op[i].at<=boundary))
	  {
if((op[i].at!=temp_at)){
at_pos=at_pos+1;
op[i].ww=at_pos;
temp_at=op[i].at;
	    }
else{
op[i].ww=at_pos;
temp_at=op[i].at;
	    }
	  }
	}
for(int i=1;i<=l;i++)
	{
if((op[i].bt>0)&&(op[i].at<=boundary))
	  {
op[i].eff = op[i].bw + op[i].ww + op[i].pw;
	  }
	}
for(int k=1;k<=l;k++)
	{
for(int i=1;i<l;i++)
	{
```

```
if((op[i+1].eff<op[i].eff)&&(op[i+1].bt>0)&&(op[i].bt>0)&&(op[i].at<=boundary)&&(op[i+1].
at<=boundary))
        {
temp=op[i+1].pri;
op[i+1].pri=op[i].pri;
op[i].pri=temp;
temp=op[i+1].at;
op[i+1].at=op[i].at;
op[i].at=temp;
temp=op[i+1].bt;
op[i+1].bt=op[i].bt;
op[i].bt=temp;
temp=op[i+1].pr;
op[i+1].pr=op[i].pr;
op[i].pr=temp;
temp = op[k+1].copy_of_bt;
op[k+1].copy_of_bt = op[k].copy_of_bt;
op[k].copy_of_bt = temp;
temp=op[i+1].bw;
op[i+1].bw=op[i].bw;
op[i].bw=temp;
temp=op[i+1].ww;
op[i+1].ww=op[i].ww;
op[i].ww=temp;
temp=op[i+1].pw;
op[i+1].pw=op[i].pw;
op[i].pw=temp;
```

```
temp=op[i+1].eff;
op[i+1].eff=op[i].eff;
op[i].eff=temp;
          }
       }
   }
for(int j=1;j<=l;j++)
   {
if((op[j].bt>0)&&(op[j].at<=boundary))
          {
             t+=op[j].bt;
response = t;
op[j].bt=0;
op[j].tat=t-op[j].at;
op[j].wt=op[j].tat-op[j].copy_of_bt;
waiting[ind]=op[j].wt;
ind++;
context = context + 1;
          }
   }
boundary=t;
}
inttot_turn=0,tot_wait=0;
for(int i=1;i<=l;i++)
   {
tot_turn+=op[i].tat;
tot_wait+=op[i].wt;
```

```
    }
floatavg_turn=(float)tot_turn/(l);
floatavg_wait=(float)tot_wait/(l);
doublevariance_result=Variance.variance(waiting)/(l);
String Anomm=String.valueOf(String.format("%.3f ",+avg_turn));
String Anom=String.valueOf(String.format("%.3f ",+avg_wait));
String Ano=String.valueOf(String.format("%.3f ",+variance_result));
Cont_Switch=l-1;
Av_TAT = Double.parseDouble(Anomm);
Av_WT = Double.parseDouble(Anom);
Vari = Double.parseDouble(Ano);
new_priority[1][l]=l;
new_priority[2][l]=Av_WT;
new_priority[3][l]=Av_TAT;
new_priority[4][l]=Vari;
new_priority[5][l]=Av_WT;
new_priority[6][l]=Cont_Switch;
}
System.out.println("New_Priority Completed");
returnnew_priority;
}
```