# Improved Dynamic Time Slice Round Robin Scheduling Algorithm with Unknown Burst Time

Bakare K. Ayeni
Department of Computer Science
Faculty of Sciences
Ahmadu Bello University,
Zaria, Nigeria

Obiniyi A. Afolayan
Department of Computer Science
Faculty of Sciences
Ahmadu Bello University,
Zaria, Nigeria

Nurat Yusuf
Department of Computer Science
Faculty of Sciences
Ahmadu Bello University,
Zaria, Nigeria

## ABSTRACT
Round-Robin (RR) is one of the algorithms mostly used in time sharing system and in network scheduling. Time slices are assigned to each process in equal portions and in circular order handling all processes without priority. Round Robin algorithm requires some parameter such as arrival time, burst time and quantum time which enables the scheduler to predict the behavior of possible processes. Prior to the execution of a process, the burst time is not known. This paper proposed a more improvement in the Round Robin CPU scheduling algorithm by improving the algorithm of Anju *et. al*. where burst time is determined using an initial time quantum. However, the improved algorithm determines burst time using *instruction count* in each of the process and by experimental analysis. This proposed algorithm performs better than Dynamic Time Slice Round Robin Scheduling Algorithm with Unknown Burst Time in terms of minimizing average waiting time, average turnaround time and number of context switches.

## Keywords
CPU scheduling algorithm, Average Waiting time, Average Turnaround Time, Number of Context Switches.

## 1. INTRODUCTION
Operating System (OS) is the brain of a computer system which constantly and continuously manages the resources available around the system in optimum way. OS controls the execution of many other application programs and acts as an interface between computer hardware and applications. It has some attractive features like multiprogramming, multitasking and multi-users, which place it way ahead in the race with human mind. One of the basic and most important tasks an OS needs to perform is job scheduling where many processing requests arrive from multiple channels to a ready queue and system manages all in a way to achieve high efficiency level.

### 1.1 CPU Scheduling
CPU scheduling is the basis of multiprogramming operating systems, by switching the CPU among processes. The operating system can make the computer more productive, whenever the CPU becomes idle; the operating system must select one of the processes in the ready queue to be executed [1].

### 1.2 CPU Scheduling Criteria and Algorithms
There are various CPU scheduling algorithms which have different properties, and the choice of a particular algorithm may favour one class of processes over another. For selection of an algorithm for a particular situation, properties of various algorithms must be considered. A good scheduling algorithm should possess the following characteristics in maximum: context switch, throughput, CPU utilization, turnaround time, waiting time and response time.

CPU scheduling algorithm:- First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling (PS) and Round Robin (RR) [1].

FCFS is the simplest form of CPU scheduling algorithm which allocates CPU to the processes on the basis of their arrival to the ready queue. Arriving processes are inserted in the tail (rear) of the ready queue and the process to be executed next is removed from the head (front) of the ready queue.

SJF, the scheduler arranged processes according to the shortest burst time in the ready queue, so that the process with least burst time is scheduled first. If two processes have equal burst time, the FCFS is applied. Long running processes may wait for prolonged periods, because the CPU has a steady supply of short processes.

PS associates each process with a priority number. The CPU is allocated to the process with the highest priority. If there are multiple processes with same priority, then FCFS will be used to allocate the CPU. Lower priority processes may starve, because the CPU may have a steady supply of higher priority process.

Round Robin (RR) scheduling algorithm is designed specifically for time-sharing systems. It is a preemptive version of first-come, first-served scheduling. Processes are dispatched in a first-in-first-out sequence but each process is allowed to run for only a limited amount of time. This time interval is known as a time-slice or quantum. It is similar to FIFO scheduling but preemption added to switches between processes [2].

## 2. LITERATURE REVIEW
Different approaches have been used to improve the performance of CPU scheduling algorithm especially in the area of Round Robin. Integer programming was used in [3] to determine time quantum that is neither too small nor too large. In [4], an algorithm that selects shortest job and assigned to CPU for a period of 1 time quantum was proposed. The proposed algorithms in [5] introduced a queue that arranges processes in ascending order of their remaining burst time in order to improve the performance of scheduling in term of Waiting Time, Turnaround Time, Response Time and Number of Context Switch. Abdulrasaq *et al* in [6] improved

on [5], Manish and Faizur algorithm by introducing another queue; arrived queue where processes are arranged in ascending order in arrive queue and time quantum is calculated by average of burst time, taking the ceiling of the result as the quantum time.

Anju *et al* in [7] proposed an algorithm to address the problem of assumption of burst time since burst time of process is not known prior to the execution of process. Their algorithm assigns initial figure as time quantum while processes execute in one cycle and burst time are determined from processes that finished with the initial time quantum.

# 3. PROPOSED ALGORITHM

Our algorithm is the modification of [7)], Dynamic Time Slice Round Robin with Unknown Burst Time (DTSRRUBT) CPU scheduling algorithm. The algorithm introduced a technique that calculates burst time using instruction count, process arrived randomly and quantum time is dynamically determined. A model that calculates burst time of processes is designed, number of processes to be executed is required to be entered with their frequency while burst time are generated for those processes and their details.

## 3.1 Illustrative Example

For example a 900MHz processor was used to execute a benchmark program with the following instruction mix and clock cycle count in table 1:

**Table 1: Instruction mix**

| S/N | Instruction Type | Instruction Count | Clock Cycle Count |
|---|---|---|---|
| 1. | Data Transfer | 1000 | 3 |
| 2. | Instructions Fetch | 1500 | 2 |
| 3. | Brach Instructions | 4000 | 2 |
| 4. | Floating Point | 4500 | 2 |
| 5. | Input/output Fetch | 8000 | 1 |
| 6. | Store Instructions | 2500 | 2 |

$\sum instruction\ count$: 21500

Burst time is calculated as thus:

$$CPI = \frac{\sum(IIC) \times (CCI)}{IC}$$

$$CPI = \frac{1000 \times 3 + 1500 \times 2 + 4000 \times 2 + 4500 \times 2 + 8000 \times 1 + 2500 \times 2}{21500}$$

$$CPI = 1.67$$

Therefore:

$$Execution\ time\ (T) = CPI \times Instruction\ Count \times Clock\ time$$

$$= \frac{CPI \times Instruction\ Count}{frequency}$$

$$T = \frac{1.67 * 21500}{900 * 21500} = \frac{1.6}{900} = 0.0018556$$

0.0018556 * 1000

Burst Time: = 1.86

However, after generating the burst time for the processes and arrival time, the algorithm takes the first process and assigns to CPU for initial time quantum of three (3), while executing the first process, it checks if other processes have arrived the ready queue, if processes arrived, it calculates average burst time of the processes as time quantum, each process then runs for the time quantum calculated, process that did not finish execution moved to the tail of the ready queue. However, it continues in the same fashion until all processes finished execution and if no process again in the ready queue, it calculates average waiting time, average turnaround time, average response time and number of context switch.

## 3.2 The pseudo code of the proposed Improved Dynamic Time Slice Round Robin CPU scheduling algorithm with Unknown Burst Time

Step 1: Start

Step 2: Enter the number of processes to be processed

Step 3: Enter the frequency of the processor

Step 4: System generates the burst time for processes with random arrival time

Step 5: WHILE (READY QUEUE!= NULL)

Step 6: If ($process\_Index == 1$)

$time_{quantum} = current\ time\ quantum$

Assign the first process

($pr[process\_Index]$) to CPU

Else

$time_{quantum} \left\lceil \frac{\sum_{i=1}^{n} burst\ time[i]}{n} \right\rceil$

Step 7: END If

Step 8: Allocate the CPU to the first process in ARRIVE queue for a period of 1 time quantum.

Step 9: If the burst time of the currently running process is remaining it moves the process to the tail of the arrive queue and if it finishes it removes the process from queue and go to step 5.

Step10: If a new process arrives the system, it is placed in the ARRIVE queue.

Step 11: END WHILE.

Step 12: Calculate AWT, ATAT, ART and NCS.

Step 13: END

## 3.3 Flow Chart

Figure 1 shows the flow chart of the proposed Dynamic Time Slice Round Robin CPU scheduling algorithm, the shaded figures are the improvement made.

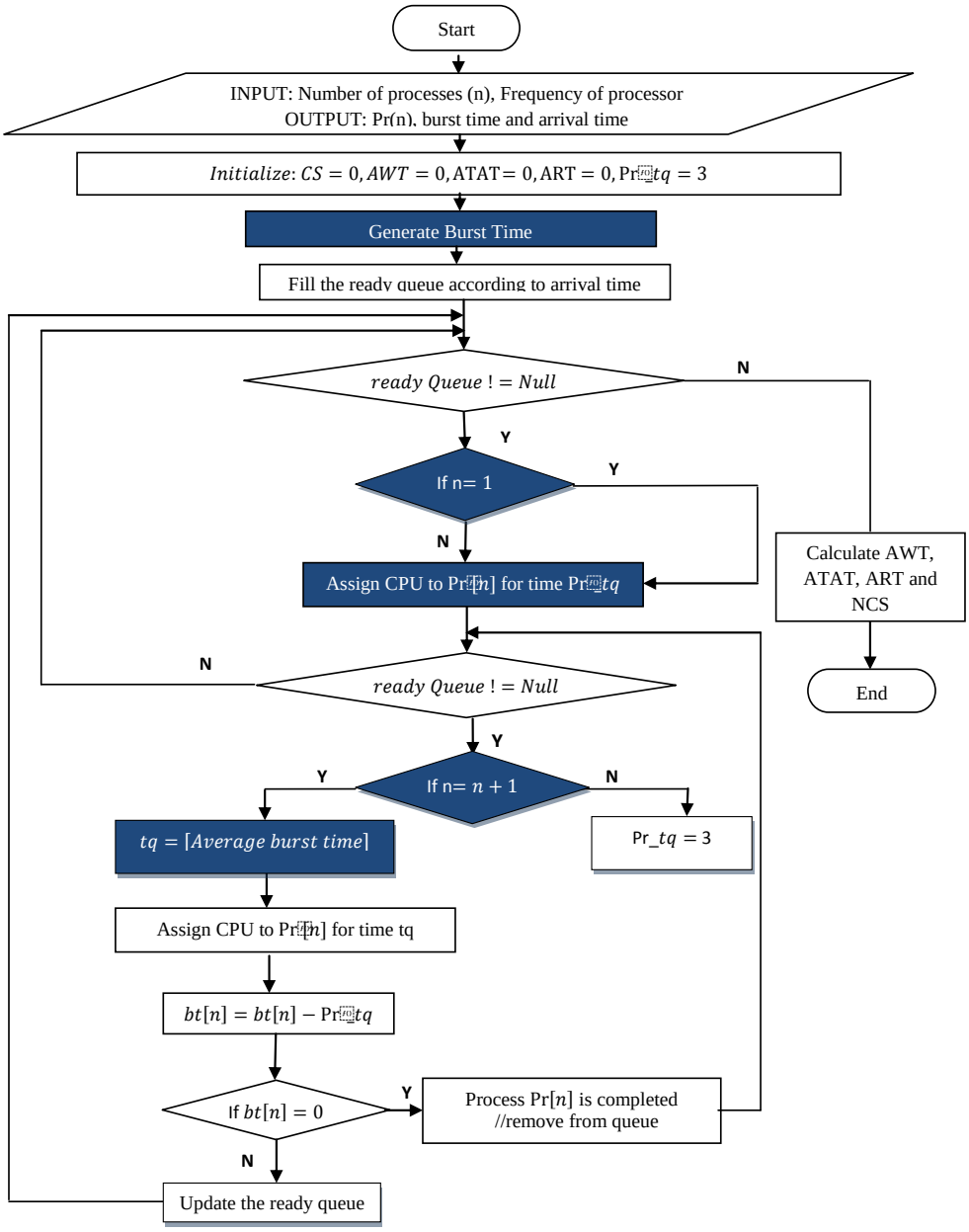


**Figure 1: The Flow Chart of the Proposed Dynamic Time Slice Round Robin CPU Scheduling Algorithm**

## 3.4 Simulation

To demonstrate the proposed algorithm and benchmark against other algorithms, table 2 below contains each process with its burst time and arrival time used for the simulation of First Come First Serve**,** Shortest Job First, Round Robin, Improved Round Robin, Dynamic Time Slice Round Robin and Improved Dynamic Time Slice Round Robin. The results are shown in table 3, table 4, table 5, table 6, table 7 and table 8 respectively.

**Table 2: Illustrative Table**

| Process ID | Arrival time (ms) | Burst time (ms) |
|---|---|---|
| P1 | 0.0 | 3.2 |
| P2 | 2.0 | 2.87 |
| P3 | 2.0 | 2.86 |
| P4 | 3.0 | 3.61 |
| P5 | 5.0 | 3.45 |

Table 3 shows simulation results for FCFS algorithm and Figure 2 shows the graph of the result from the table, from the graph the response time and the waiting time is high but the context switch is low since the process switched once. Table 4 shows simulation results for SJF algorithm and Figure 3 shows the graph of the result from the table, the performance is better compared to FSCS in term of AWT and ART since priority is given to the processes with short burst time. Table 5 shows simulation results for conventional RR along with the graph in Figure 4. Table 6 shows the simulation results for IRR along with the graph in Figure 5. Table 7 is the results of simulation for DTSRR and Figure 6 shows the graph of DTSRR and Table 7 is the results of proposed algorithm simulated along with the graph shows in Figure 7. However, the performance of the three round robin algorithms when compared, IDTSRR is better. All the graphs are graph of X and Y axis, where the X-axis represents the time and Y-axis represents the process ID.

**Table 3: First Come First Serve**

| Process | Arrival | Burst | Start | Finish | Waiting | Turnaround | Response | Context |
|---|---|---|---|---|---|---|---|---|
| P1 | 0.0 | 3.2 | 0.0 | 3.2 | 0.0 | 3.2 | 0.0 | 1 |
| P2 | 2.0 | 2.87 | 3.2 | 6.07 | 1.2 | 4.07 | 1.2 | 1 |
| P3 | 2.0 | 2.86 | 6.07 | 8.93 | 4.07 | 6.93 | 4.07 | 1 |
| P4 | 3.0 | 6.61 | 8.93 | 12.54 | 5.93 | 9.54 | 5.93 | 1 |
| P5 | 5.0 | 3.45 | 12.54 | 15.99 | 7.54 | 10.99 | 7.54 | 1 |
| Average | | | | | 3.75 | 6.95 | 3.75 | 5 |

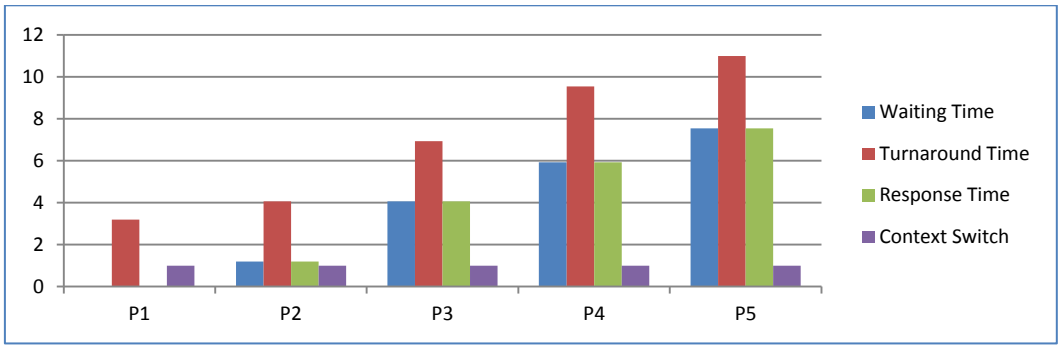


**Figure 2: FCFS Graph**

**Table 4: Shortest Job First**

| Process | Arrival | Burst | Start | Finish | Waiting | Turnaround | Response | Context |
|---|---|---|---|---|---|---|---|---|
| P1 | 0.0 | 3.2 | 0.0 | 3.2 | 0.0 | 3.2 | 0.0 | 1 |
| P3 | 2.0 | 2.86 | 3.2 | 6.06 | 1.2 | 4.06 | 1.2 | 1 |
| P2 | 2.2 | 2.87 | 6.06 | 8.93 | 4.06 | 6.93 | 4.06 | 1 |
| P4 | 3.0 | 3.61 | 12.38 | 15.99 | 9.38 | 12.99 | 9.38 | 1 |
| P5 | 5.0 | 3.45 | 8.93 | 12.38 | 3.93 | 7.38 | 3.93 | 1 |
| Average | | | | | 3.71 | 6.27 | 3.71 | 5 |

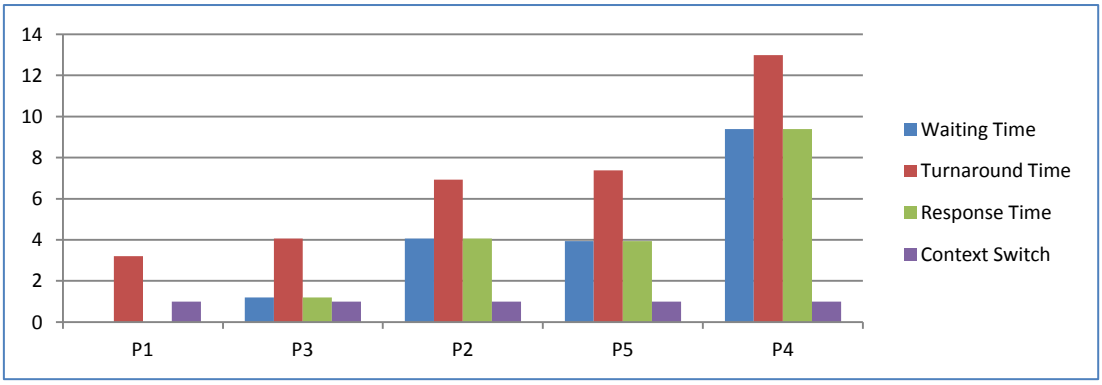


**Figure 3: SJF Graph**

**Table 5: Round Robin**

| Process | Arrival | Burst | Start | Finish | Waiting | Turnaround | Response | Context |
|---|---|---|---|---|---|---|---|---|
| P1 | 0.0 | 3.2 | 0.0 | 7.2 | 4.0 | 7.2 | 0.0 | 2 |
| P2 | 2.0 | 2.87 | 2.0 | 10.07 | 5.2 | 8.07 | 0.0 | 2 |
| P3 | 2.0 | 2.86 | 4.0 | 12.93 | 8.07 | 10.93 | 2.0 | 2 |
| P4 | 3.0 | 3.61 | 7.2 | 14.54 | 7.93 | 11.54 | 4.2 | 2 |
| P5 | 5.0 | 3.45 | 10.07 | 15.99 | 7.54 | 10.99 | 5.07 | 2 |
| Average | | | | | 6.55 | 9.75 | 2.25 | 10 |

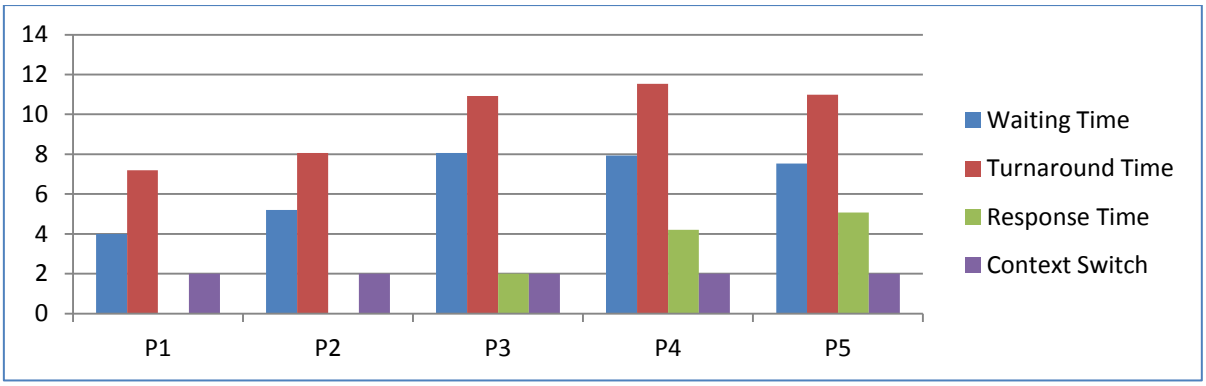


**Figure 4: Round Robin Graph**

**Table 6: Improved Round Robin**

| Process | Arrival | Burst | Start | Finish | Waiting | Turnaround | Response | Context |
|---|---|---|---|---|---|---|---|---|
| P1 | 0.0 | 3.2 | 0.0 | 3.2 | 0.0 | 3.2 | 0.0 | 1 |
| P2 | 2.0 | 2.87 | 3.2 | 6.07 | 1.2 | 4.07 | 1.2 | 1 |
| P3 | 2.0 | 2.86 | 6.07 | 8.93 | 4.07 | 6.93 | 4.07 | 1 |
| P4 | 3.0 | 3.61 | 8.93 | 15.54 | 8.93 | 12.54 | 5.93 | 2 |
| P5 | 5.0 | 3.45 | 11.93 | 15.99 | 7.54 | 10.99 | 6.93 | 2 |
| Average | | | | | 4.35 | 7.55 | 3.63 | 7 |

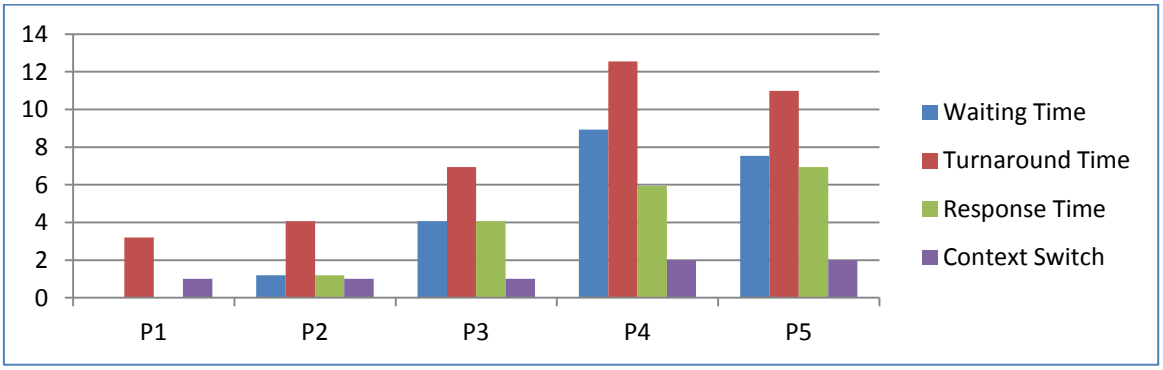


**Figure 5: Improved Round Robin Graph**

**Table 7: Dynamic Time Slice Round Robin**

| Process | Arrival | Burst | Start | Finish | Waiting | Turnaround | Response | Context |
|---|---|---|---|---|---|---|---|---|
| P1 | 0.0 | 3.2 | 0.0 | 7.7 | 4.5 | 7.7 | 0.0 | 2 |
| P2 | 2.0 | 2.87 | 2.0 | 10.57 | 5.7 | 8.57 | 0.0 | 2 |
| P3 | 2.0 | 2.86 | 4.5 | 13.43 | 8.57 | 11.43 | 2.5 | 2 |
| P4 | 3.0 | 3.61 | 7.7 | 14.54 | 7.93 | 11.54 | 4.7 | 2 |
| P5 | 5.0 | 3.45 | 10.57 | 15.49 | 7.04 | 10.49 | 5.57 | 2 |
| Average | | | | | 6.75 | 9.95 | 2.55 | 10 |

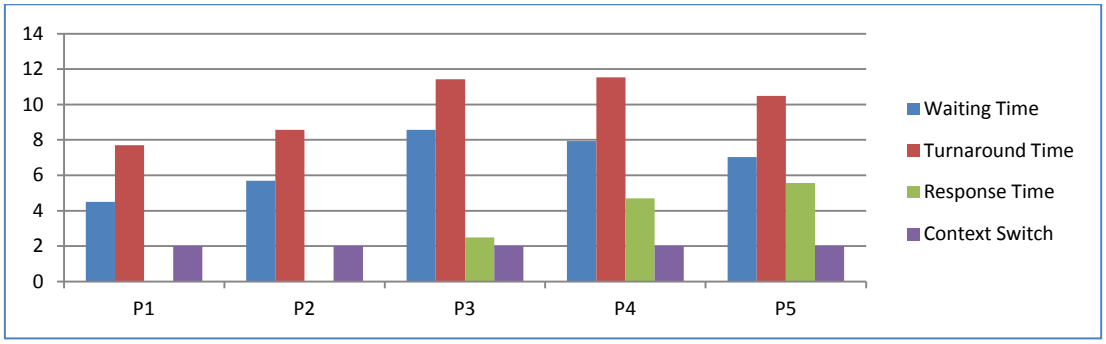


**Figure 6: DTSRR Graph**

**Table 8: Improved Dynamic Time Slice Round Robin**

| | Arrival | Burst | Start | Finish | Waiting | Turnaround | Response | Context |
|---|---|---|---|---|---|---|---|---|
| P1 | 0.0 | 3.2 | 0.0 | 12.54 | 9.34 | 12.54 | 0.0 | 2 |
| P2 | 2.0 | 2.87 | 3.0 | 5.87 | 1.0 | 3.87 | 1.0 | 1 |
| P3 | 2.0 | 2.86 | 5.87 | 8.73 | 3.87 | 6.73 | 3.87 | 1 |
| P4 | 3.0 | 3.61 | 8.73 | 12.34 | 5.73 | 9.34 | 5.73 | 1 |
| P5 | 5.0 | 3.45 | 12.54 | 15.99 | 7.54 | 10.99 | 7.54 | 1 |
| Average | | | | | 5.5 | 8.69 | 3.63 | 6 |

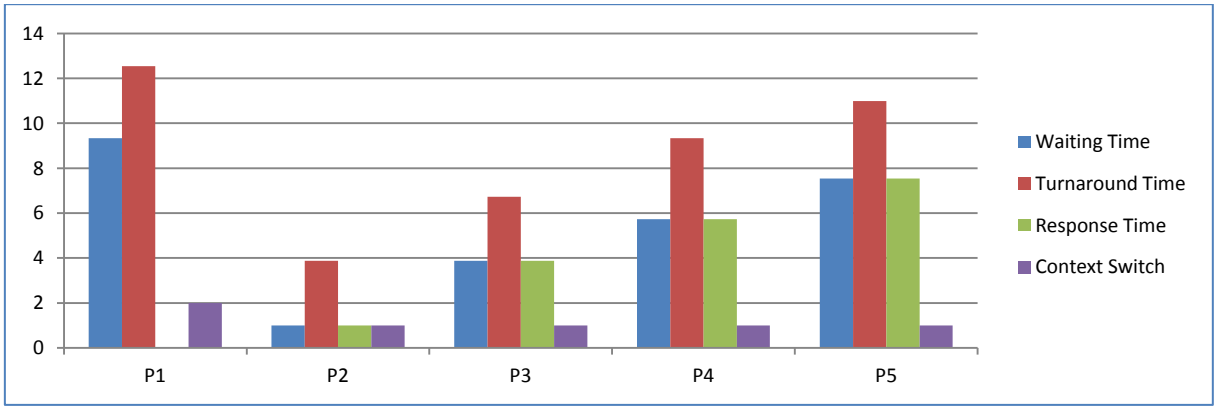


**Figure 7: Improved Dynamic Time Slice Round Robin Graph**

## 3.5 Comparative Analysis

To compare the performance of the six algorithms, the results in Table 9 is the evaluation of performance using five processes for all the six algorithms, one hundred processes were used in Table 10 and One thousand processes was used in Table 11 respectively. The graphs of evaluation are shown in Figure 8, Figure 9 and Figure 10 respectively.

**Table 9: Comparative table using 5 processes**

| Algorithms | AWT | ATAT | ART | NCS |
|---|---|---|---|---|
| **FCFS** | 3.75 | 6.95 | 3.75 | 5 |
| **SJF** | 3.71 | 6.27 | 3.71 | 5 |
| **RR** | 6.55 | 9.75 | 2.25 | 10 |
| **IRR** | 4.35 | 7.55 | 6.63 | 7 |
| **DTSRRUBT** | 6.75 | 9.95 | 2.55 | 10 |
| **IDTSRRUBT** | 5.5 | 8.69 | 3.63 | 6 |

**Table 10: Comparative Table using 100 Processes**

| Algorithms | AWT | ATAT | ART | NCS |
|---|---|---|---|---|
| **FCFS** | 132.87 | 136.56 | 132.87 | 100 |
| **SJF** | 111.17 | 114.83 | 111.17 | 100 |
| **RR** | 211 | 214.68 | 80.83 | 322 |
| **IRR** | 205.88 | 209.57 | 108.24 | 182 |
| **DTSRRUBT** | 206.43 | 210.12 | 95.28 | 200 |
| **IDTSRRUBT** | 177.69 | 181.37 | 122.78 | 144 |

**Table 11: Comparative Table using 1000 Processes**

| Algorithms | AWT | ATAT | ART | NCS |
|---|---|---|---|---|
| **FCFS** | 1332.29 | 1335.96 | 1332.29 | 1000 |
| **SJF** | 1097.3 | 1100.97 | 1097.3 | 1000 |
| **RR** | 2091.51 | 2095.18 | 791.95 | 2322 |
| **IRR** | 2029.7 | 2033.37 | 1066.82 | 1781 |
| **DTSRRUBT** | 2082.51 | 2086.19 | 933.78 | 2006 |
| **IDTSRRUBT** | 1748.56 | 1752.24 | 1221.44 | 1411 |

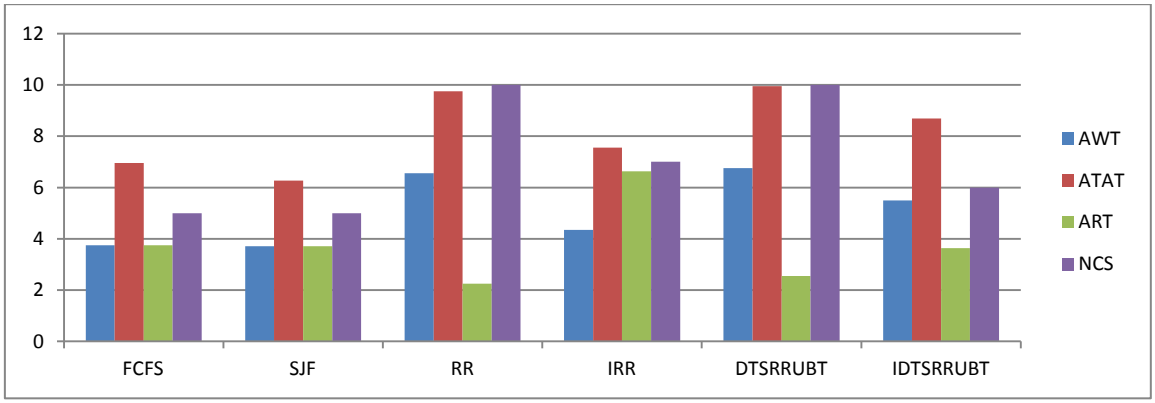


**Figure 8: Graph of Evaluation Criteria using 5 Processes**

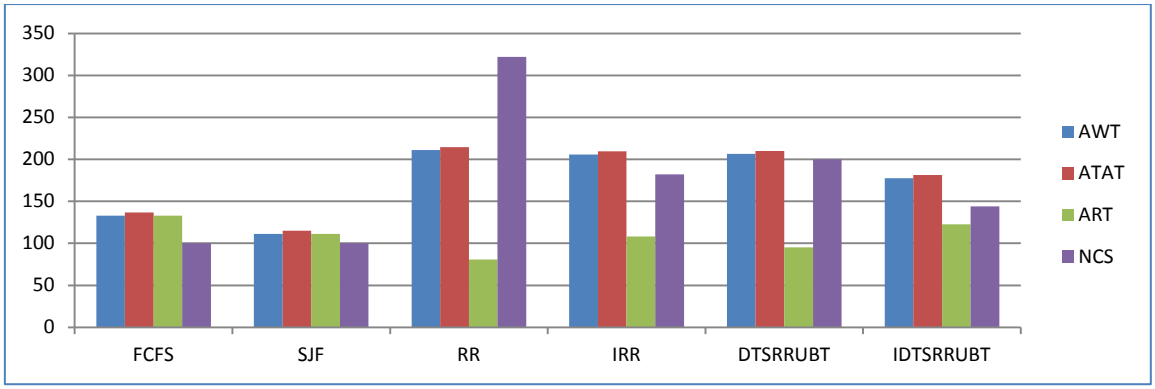


**Figure 9: Graph of Evaluation Criteria using 100 Processes**

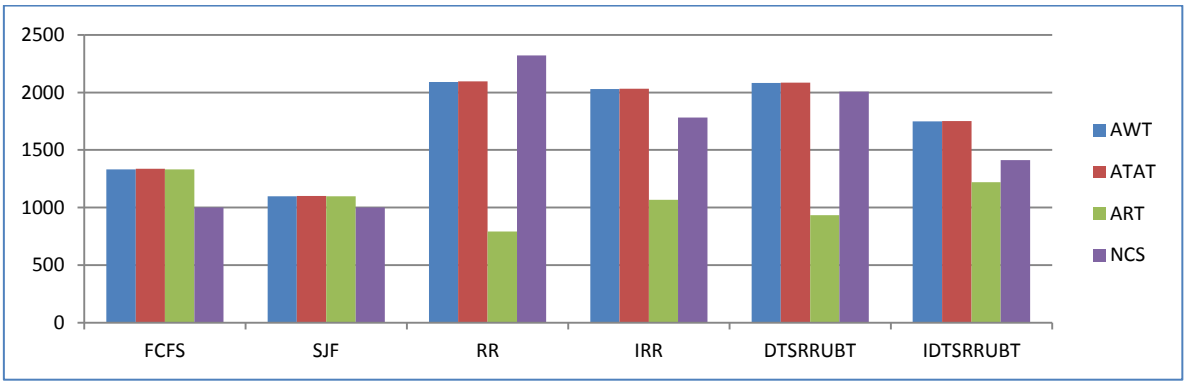


**Figure 10: Graph of Evaluation Criteria using 1000 Processes**

From the study of comparative tables and graphs, it was discovered that the Improved Dynamic Time Slice Round Robin with unknown burst time (IDDTSRR) is better in term of average waiting time, average turnaround time and context switch. Apparently, if there are millions of processes the algorithm will still performed better. It is desirable to minimize all the four performance criteria. The proposed algorithm when compared with the existing algorithms minimized the three out of the four criteria which makes it better than the existing one.

## 4. CONCLUSION

The proposed algorithm was presented to enhance dynamic time slice round robin with unknown burst time scheduling algorithm. The algorithm determines burst time using instruction count in each of the process by experimental analysis. The simulation results showed that this approach minimized average waiting time, average turnaround time and number of context switches.

Future research should focus on investigating other method of determining burst time of processes to improve efficiency of scheduling techniques.

## 5. REFERENCES

[1] Silberschatz, A., Galvin, P. B. and Gagne, G. 2013. Operating System Concepts. (9th, Ed.) John Wiley and Sons Inc, USA.

[2] Ajit, S, Priyanka, G and Sahil, B. 2010. An Optimized Round Robin Scheduling Algorithm for CPU Scheduling. *International Journal on Computer Science and Engineering* , *2* (7), 2382-2385

[3] Hamad, S. H., Mostafa, S., and Rida, S. Z. 2010 Finding Time Quantum of Round Robin CPU Scheduling Algorithm in General Computing Systems using Integer Programming. IJRRAS, 5(1) pp 65-70.

[4] Nirvikar And Kumar, N. 2013 Performance Improvement Using CPU Scheduling Algorithm-SRT. International *Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, (2) 2. Pp 110-113.

[5] Manish, K. M. and Faizur R. 2014 An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum. International Journal of Computer Science, Engineering and Applications (IJCSEA) (4)4.

[6] Abdulrazak, A., Abdullahi, S. E. and Sahalu, J. B. 2014. New Improved Round Robin (NIRR) CPU Scheduling Algorithm. *International Journal of Computer Applications* (0975 – 8887) Volume 90 – No 4, March 2014.

[7] Anju, M., Neenu, A., and Nandakumar, R 2016. Dynamic Time Slice Round Robin Scheduling Algorithm with Unknown Burst Time. *Indian Journal of Science and Technology*, Vol 9(8), pp. 1-6.