International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

# COMPARATIVE PERFORMANCE AND ANALYSIS OF SOME IMPROVED ROUND ROBIN CPU SCHEDULING

K. I Musa, K. E Lasisi, J.A Gokir

Abstract

Some researchers in recent times still compare their proposed algorithms with the famous algorithm such as FCFS, SJF RR when there are already modified in the research domain. Thus, the paper seeks to find whether any latest improvement implies improvement on the improved, assuming that all immediate preceding (improved) algorithms have been studied, therefore a need to establish a research ladder in the field of Round Robin improvement to come out with the most optimal of all improved algorithms for implementation in time sharing and real time operating system with time. For this purpose, the researchers randomly selects the algorithm (DABRR) proposed in 2015 and two (DTSRR and RMRR) of 2016, studied the algorithms, implement the algorithms using VB 2013 and their analyzed percentage result presented in graphs with the aid of spreadsheet. The result showed that DABRR proposed in 2015 performs more optimal in comparison to those (DTSRR and RMRR) of 2016 since the analyzed result proofs that DABRR improves traditional RR by 37.13%, 36.75% and 98.49%, DTSRR improves traditional RR by 7.06%, 7.04%, 86.36% and RMRR improves traditional RR by17.04%, 12.75% and 97.65%, each with respect to AWT, ATAT and NCS respectively. Thus, the research proofs that not all latest proposal are actual improvement on the previous proposals therefore the recommendation that researchers should always compare their new algorithms with already improved algorithms not to meander within by rewinding back to the famous algorithms. Considering hierarchical (ladder) performance classifications of improvements on RR algorithm, DABRR tops the RMRR followed by DTSRR and the RR

**Index Terms**: Comparatives Performance, Comparative Analysis, Round Robin CPU Scheduling, Analysis of Improved Round Robin

## 1. INTRODUCTION

Considering the fact that the number of processes loaded into computer memory ready for execution at a time may by more than the number of processors in the system where the processor is left with a diverse choice of which process be executed first?, for how long?, which next?, what are the subsequent order of executions? What is the turn for each process? However, with the aid of a scheduling concept all the above questions were addressed. Based on this concept **[1]** affirmed that in a single processor system, only one process can be allowed to run at a time. Similarly [2] stated that only one process runs a time in uni-processor system and any other must wait until CPU is free and can be allowed. In a case where the number of processor is more than or equal to number of programs loaded, still there are choices of which processor executes which program or which processor be kept busy and which be kept idle must be made **[3].** Therefore, these called for a mechanism or program with a conflict settling policy that controls these choices, order of execution and execution duration. A program that settles conflicts among processes by determining their order and duration of execution can be referred to as a scheduler and the process of performing the function is named scheduling. Thus the definition with respect to CPU by **[4]** that CPU scheduling is a decision of allocating a single resource among multiple clients, the order of allocation and durations. In order to be convinced that this scheduling is not just common with CPU**, [5]** listed areas where scheduling are evidently applied in real life as road, railway and air traffic control system and league games.

This scheduling concept was implemented via algorithm called scheduling algorithm, thus the following examples: First Come First Serve (FCFS), Shortest Job First (SJF), and

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

Round Robin (RR). These algorithms were designed based on suitability of implementations compatible with the behaviour of the system that achieved its objectives. It is in this light that **[6]** listed algorithms with their suitable respective areas of application as: FCFS and SJF are suitable for batch system and non interactive while RR suitable for time-shared and interactive systems. Each of the algorithms is expected to exhibit a good quality of maintaining a minimum turnaround time, waiting time, response time, context switch and maximum throughput and CPU utilization irrespective of its system of application **[1] and [7]**

In a quest to realized algorithms with optimal quality with respect to the above listed metrics, researchers such as **[1]** , **[4], [5], [7]** etc are always on their feet to see to the optimal performance of RR algorithms. However, with the calibre of theses researchers, one can assumed that RR algorithm should be at the optimal level not deserving improvement any longer but it seems that most of the researchers with respect to improving RR algorithms worked in parallel mode without harmonization by failure to compare their proposed algorithms performance with respect to the preceded improved RR, instead compared theirs with the famous RR, FCFS and SJF, however, only few were harmonized. This was evident in the researches of [8**], [9], [10]** and **[11].**This was also observed by **[12]** that rarely have larger number of improved RR algorithms compared with those just been improved by other researchers. Worthy of commendation for comparing the performance of their proposed RR algorithms with previous improvement were **[13],** [14], [15]**, [1] and [16]**

Nevertheless, a step in the ladder of RR optimization should be the order of the day by comparing the performance of the existing un-compared algorithms to ascertain their performance before thinking of proposing new and the most optimized of all otherwise on may be proposed an algorithm which may perform less than some that had earlier been proposed. Some of the researchers who strictly compared algorithms proposed by others without proposing theirs were **[17]), [18], [19] and [12].**

It is in this light that this paper compares the analyzed performance of three improved RR algorithms to confirm the optimality performance between precedes and succeeds algorithms. Therefore, the researchers considered the DABRR of **[14],** RMRR of **[11]** and DTSRR of **[15].** The famous RR was used in the research as a standard point for comparing the improved (optimized) RRs. These algorithms were randomly selected among the proposed algorithms in recent times (year 2015-2016). Critical to RR operation and optimization are the TQ computation model, TQ regulation and allocation of resource duration, process arrangement and execution order. The application of the parameters can be seen in the considered algorithms reviewed below.

### 1.1 DABRR Algorithm

**[14]** In a research titled An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum proposed a dynamic Average Burst Round Robin with $TQ = \frac{\sum \text{Burst times}}{n}$ implemented execution order in cases. Case I sort processes in increasing order of burst times, Case ii sort processes in decreasing order of burst times and case iii execute processes in order of arrival FCFS out of all, case i proofs to be the most optimal. DABRR maintained maximum of one TQ to each process at a time in a round of execution after which the processor is relocated to the next process in the ready queue. The DABRR was implemented alongside with RR, DQRRRR, IRRVQ, SARR, RP_5, and MRR for comparison and analysis.

### 1.2 RMRR

**[11]** Proposed A Revamped Mean Round Robin (RMRR) CPU Scheduling Algorithm with TQ computation model $TQ = \frac{\sum \text{Burst times}}{n}$ that execute processes in FCFS order. RMRR regulates its TQ for allocation and de-allocation of resources as thus, If RBT<TQ then execute the process to completion else executed one TQ period and allocate the processor to the next process on the ready queue. RMRR was implemented alongside with RR, FCFS, & SJF for comparative performance

### 1.3 DTSRR Algorithm

**[15]** Proposed a Dynamic Time Slice Round Robin (DTSRR) Scheduling Algorithm With Unknown Burst Time which execute processes in order of their arrival times (FCFS). First of all, TQ is arbitrary Selected and subsequently

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

updated using number of processes that finishes in a round of execution as thus, If Finish<1 then TQ=TQ*2 else if finish>2 then$TQ = \frac{TQ}{2}$. If number of processes that Finish in a round of execution is less than one then increment the TQ by TQ=TQ*2 else if number of processes that finishes in a round of execution is greater than two then decrement the TQ by $TQ = \frac{TQ}{2}$ else maintain the TQ. The DTSRR was implemented with RR and Optimized RR for performance comparison.

**2 ILLUSTRATIVE EXAMPLE**

For the purpose of evaluation, the equation used in obtaining waiting time and average waiting time for each scheduling algorithm are giving as follows:
The equation used in obtaining turnaround time and average turnaround time for each scheduling algorithm is giving as follows:
Turnaround time
(TAT)=Finished time (FT)-Arrival time (AT)
TAT=FT-AT------i
Average turnaround time
(ATAT)=$\frac{\text{Sum of all processes turnaround time}}{\text{Number of processes}}$----ii
Waiting time (WT) =TAT-burst time (BT) i.e.
WT=TAT-BT-----iii
Average waiting time
(AWT)=$\frac{\text{Sum of all processes waiting time}}{\text{Number of processes}}$-----iv
ATAT % increment=$\frac{(\text{RRGranttotal}-\text{improved Granttotal})*100}{\text{RR(ATAT)Granttotal}}$--v

AWT % increment=$\frac{(\text{RRGranttotal}-\text{improved Granttotal})*100}{\text{RR(AWT)Granttotal}}$---vi

NCS % increment=$\frac{(\text{RRGranttotal}-\text{improved Granttotal})*100}{\text{RR(NCS)Granttotal}}$--vii
In order to bring the performance metrics and the considered algorithms performance to bare, even before the simulation, the following example of randomly generated processes in table 1.00 were considered.

Table 4.0: Randomly generated processes

| Process | burst time | Arrival Time |
|---|---|---|
| P1 | 12 | 0 |
| P2 | 10 | 0 |
| P3 | 24 | 0 |
| P4 | 16 | 0 |
| P5 | 9 | 0 |

Since DABRR demands sorting of processes in ascending order and table 1.1 handles that as shown below
Table 1.1: Processes Sorted in ascending order of burst time

| Process | burst time | Arrival Time |
|---|---|---|
| P5 | 9 | 0 |
| P2 | 10 | 0 |
| P1 | 12 | 0 |
| P4 | 16 | 0 |
| P3 | 24 | 0 |

Algorithm RR, DTSRR and RMRR implements table 1.0 since they executes in FCFS while DABRR executes table 1.1 since it has to sort the ready processes in ascending order of burst times.

| P1 | P2 | P3 | P4 | P5 | P1 | P2 |
|---|---|---|---|---|---|---|

0 5 10 15 20 25 30 35

| P3 | P4 | P5 | P1 | P3 | P4 | P3 |
|---|---|---|---|---|---|---|

40 45 49 51 56 61 66

| P4 | P3 |
|---|---|

67 71

Figure 1.0: RR Gantt Chart

ATAT=$\frac{(51-0)+(35-0)+(71-0)+(67-0)+(49-0)}{5}$=54.6
AWT=$\frac{(51-12)+(35-10)+(71-24)+(67-16)+(49-9)}{5}$=40.4
NCS=15

| P5 | P2 | P1 | P4 | P3 | P4 | P3 |
|---|---|---|---|---|---|---|

0 9 19 31 45 59 61 71

Figure 1.1: DABRR Gantt Chart

ATAT=$\frac{(9-0)+(19-0)+31-0)+(61-0)+(71-0)}{5}$= 38.2

AWT=$\frac{(9-9)+(19-10)+(31-12)+(61-16)+(71-24)}{5}$= 24
NCS = 6

| P1 | P2 | P3 | P4 | P5 | P1 |
|---|---|---|---|---|---|

0 5 10 15 20 25 32

| P2 | P3 | P4 | P5 | P3 | P4 | P3 |
|---|---|---|---|---|---|---|

37 47 57 61 66 67 71

Figure 1.2: DTSRR Gantt Chart

ATAT=$\frac{(32-0)+(37-0)+(71-0)+(67-0)+(61-0)}{5}$=53.6
AWT=$\frac{(32-12)+(37-10)+(71-24)+(67-16)+(61-9)}{5}$ =39.4
NCS=12

| P1 | P2 | P3 | P4 | P5 | P1 | P2 |
|---|---|---|---|---|---|---|

0 2 4 6 8 10 20 28

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

| P3 | P4 | P5 | P3 | P4 |
|---|---|---|---|---|
| 40 | 52 | 59 | 69 | 71 |

Figure 1.3: RMRR Gantt Chart

TQ=2.12

ATAT=$\frac{(20-0)+(28-0)+(69-0)+(71-0)+(59-0)}{5}$=49.4

AWT=$\frac{(20-12)+(28-10)+(69-24)+(71-16)+(59-9)}{5}$=35.2

NCS= 11

Table 1.2: Performance Metrics of considered algorithms

| METRICS | RR | DABRR | DTSRR | RMRR |
|---|---|---|---|---|
| TQ | 5 | 14 | 5,10,5 | 2,12 |
| ATAT | 54.6 | 38.2 | 53.6 | 49.4 |
| AWT | 40.4 | 24 | 39.4 | 35.2 |
| NCS | 15 | 6 | 12 | 11 |

The above table 1.2 showed that DABRR designed in 2015 outperform DTSRR and RMRR of 2016 with respect to all the considered measuring metrics.

DABRR ATAT increment=$\frac{(54.6-38.2)*100}{54.6}$=30.03%

DABRR AWT increment=$\frac{(40.4-24)*100}{40.4}$=40.59%

DABRR NCS increment=$\frac{(15-6)*100}{15}$=60%

DTSRR ATAT increment=$\frac{(54.6-53.6)*100}{54.6}$=1.83%

DTSRR AWT increment=$\frac{(40.4-39.4)*100}{40.4}$=2.48%

DTSRR NCS increment=$\frac{(15-12)*100}{15}$=20%

RMRR ATAT increment=$\frac{(54.6-49.4)*100}{54.6}$=9.52

RMRR AWT increment=$\frac{(40.4-35.2)*100}{40.4}$=12.87%

RMRR NCS increment=$\frac{(15-11)*100}{15}$=26.67%

Table 1.3: Performance Analysis improvement of DABRR,DTSRR and RMRR

| Metrics | RR | DABRR | DTSRR | RMRR |
|---|---|---|---|---|
| ATAT | 0.00% | 30.03% | 1.83% | 9.52% |
| AWT | 0.00% | 40.59% | 2.48% | 12.87% |
| NCS | 0.00% | 60% | 20% | 26.67% |

## 3. PROCESS GENERATION AND METRICS

Processes were generated based on the activity case with respect to activity algorithm big oh analysis orders. N represents the size of the activity to be performed shown in table 1.4 below

Table 1.4: Process generation means.

| Process activity type | Activity case | | |
|---|---|---|---|
| | Best case | Average case | Worst case |
| Linear search | O(1) | O(N) | O(N) |
| Binary search | O(1) | O(logN) | O(logN) |
| Burble sort | O(N) | $O(N^2)$ | $O(N^2)$ |
| Selection sort | $O(N^2)$ | $O(N^2)$ | $O(N^2)$ |
| Insertion sort | O(N) | $O(N^2)$ | $O(N^2)$ |
| Merge sort | O(NlogN) | O(NlogN) | O(NlogN) |
| Quick sort | O(NlogN) | O(NlogN) | $O(N^2)$ |
| Heap sort | O(NlogN) | O(NlogN) | O(NlogN) |

## 4. PROCESS SIMULATION AND METRICS

Figure 1.4 below shows the Algorithm metrics for 100 processes executed. Burst times are generated with algorithm size 100 to 600. Meaning a process can have a maximum burst time of $600^2$=360000ms e.g. take the worst case or average case of burble sort, selection sort and insertion sort with $O(N^2)$ where .DTSRR and RR use TQ=250 and performance metrics as displayed in figure1.4 below.

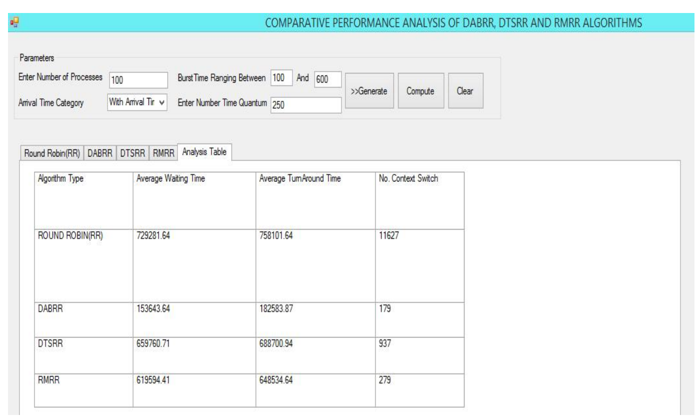


Figure: 1.4: Metrics for 100 generated and computed processes

## 5. COMPARATIVE PERFORMANCE

The table 1.5 below was a summary of metrics performance with respect to average turnaround time derived from

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

100,200, 300 and 400 processes generated and executed by RR, DABRR, DTSRR and RMRR used in obtaining figure 1.5 below. The table showed that DABRR performs better than RMRR since it has minimum figure in comparison to others, followed by RMRR and DTSRR.

**Table 1.5:** Comparative performance of algorithms' ATAT

| Processes generated | RR | DABRR | DTSRR | RMRR |
|---|---|---|---|---|
| 100 | 768101.6 | 182583.9 | 688700.9 | 648534.6 |
| 200 | 3272768 | 2624847 | 2932548 | 2856717 |
| 300 | 4253538 | 2997054 | 3858134 | 3694014 |
| 400 | 5131502 | 2687891 | 5000770 | 4515120 |

The figure 1.5 presents the graph of the data contained in table 1.5 clearly showing that DABRR had optimal ATA in comparison to DTSRR and RMRR as can also be seen in the table1.5 above.

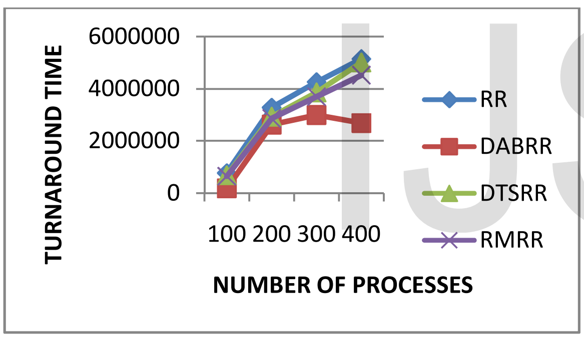


Figure 1.5 : Comparative performance of algorithms' ATAT

The table 1.6 below was a summary of simulated metric performance with respect to AWT derived from 100,200, 300 and 400 processes generated and executed by RR, DABRR, DTSRR and RMRR used in obtaining figure 1.6. The AWT table1.6 clearly showed that DABRR perform better than RMRR and DTSRR since it has minimum figure in comparison to all followed by RMRR and DTSRR

TABLE 1.6 : Comparative performance of algorithms' AWT

| Pro. Gen. | RR | DABRR | DTSRR | RMRR |
|---|---|---|---|---|
| 100 | 729281.64 | 153643.64 | 659760.71 | 61959.41 |
| 200 | 3228954.58 | 2580890.25 | 2888591.08 | 2812759.98 |
| 300 | 4211254.38 | 2954618.19 | 3815698.38 | 3651578.16 |
| 400 | 5090820.82 | 2647022.96 | 4959901.82 | 4474252.39 |

The figure 1.6 presents the graph of the data contained in table 1.6 clearly showing that DABRR had optimal AWT in comparison to DTSRR and RMRR as can also be seen in the table 1.6 above.

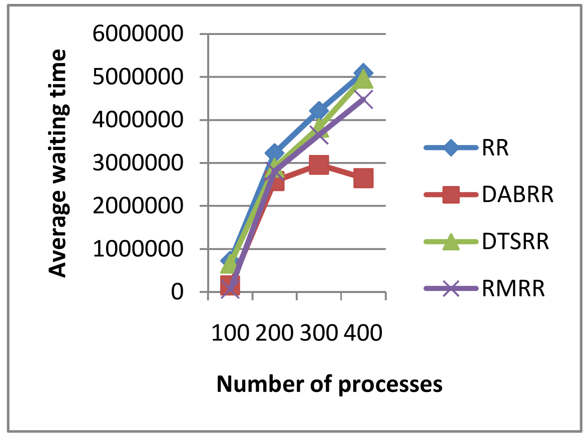


Figure 1.6 : Comparative performance of algorithms' AWT

The table 1.7 below was a summary of metrics performance with respect to NCS derived from 100,200, 300 and 400, processes generated and executed by RR, DABRR, DTSRR and RMRR used in obtaining figure 1.7. The NCS table1.6 clearly showed that DABRR perform better than DTSRR and RMRR since it has minimum figure in comparison to all followed by RMRR and DTSRR.

Table 1.7 : Comparative performance of algorithms' NCS

| NUMBER OF PROCESSES | RR | DABRR | DTSRR | RMRR |
|---|---|---|---|---|
| 100 | 11627 | 179 | 937 | 279 |
| 200 | 29408 | 356 | 2632 | 556 |
| 300 | 36542 | 534 | 2848 | 834 |
| 400 | 41080 | 720 | 9764 | 1120 |

The figure 1.7 presents the graph of the data contained in table 1.7 clearly showing that DABRR had optimal NCS in comparison to DTSRR and RMRR as can also be seen in the table 1.7 above.

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

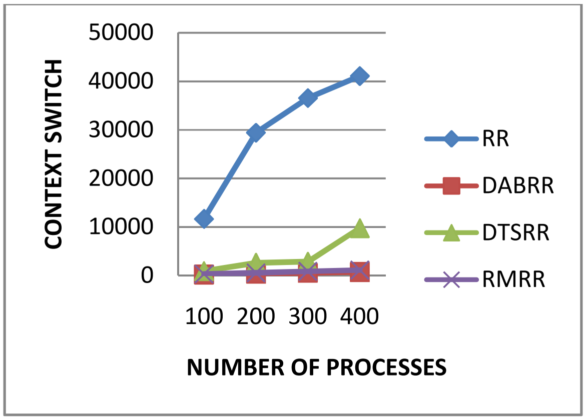


Figure1.7: Comparative performance of algorithms' Number of Context Switch (NCS)

## 6. ANALYZED COMPARATIVE PERFORMANCE

The table 1.8 below was a summary of Comparative Performance Result **Analyzed** with respect to AWT ATAT and NCS derived from table 1.5, 1.6 and 1.7.the table evidently showed that DABRR improved RR by 37.13%, 36.75% and 98.49%, DTSRR improved RR by 7.06%, 7.04% and 86.36% while RMRR improved RR by 17.04%, 12.75% and 97.65% and each with respect to AWT, ATAT and NCS respectively.

Table 1.8: Analyzed performance percentage improvement over RR

| Metric | | RR | DABRR | DTSRR | RMRR |
|---|---|---|---|---|---|
| AWT | Grant Total | 13260311 | 8336175 | 12323952 | 11000550 |
| | % improved | 0.00% | 37.13% | 7.06% | 17.04% |
| ATAT | Grant Total | 13425909 | 8492376 | 12480153 | 11714386 |
| | % improve | 0.00% | 36.75% | 7.04% | 12.75% |
| NCS | Grant total | 118657 | 1789 | 16181 | 2789 |
| | % improve | 0.00% | 98.49% | 86.36% | 97.65% |

The figure 1.8 below presents the graph of the data contained in table 1.8 clearly showing that DABRR had optimal AWT, ATAT and NCS in comparison to DTSRR and RMRR as can also be seen in the table 1.8 above.

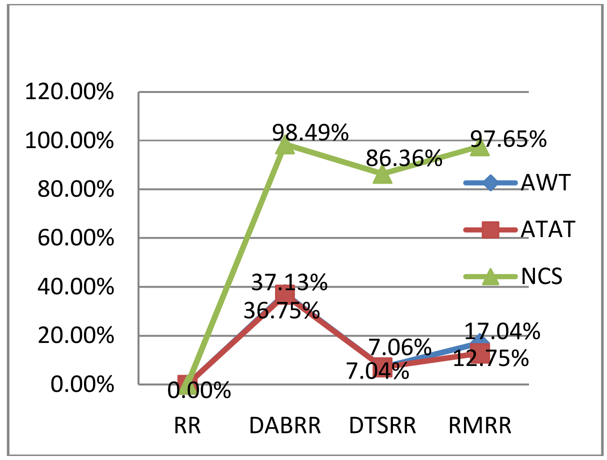


Figure 1.8: AWT, ATAT and NCS Analyzed performances of algorithms

## 7. DISCUSSION OF RESULTS

Observation from data presented in table 1.5, 1.6 and 1.7 represented figure 1.5, 1.6 and 1.7 respectively showed that DABRR performed better than RMRR and RMRR better than DTSRR in agreement to presentations format of: [11] in their table 2, graph 1, graph 2 and graph 4, [14] in their table 13 and 14, figure 4 and 5, [15] in their table 2, 4 and 6, figure 3, 4 and 5.

The analyzed result showed that DABRR is the most optimal of all with respect to all the considered metrics. It evidently improved RR by 37.13%, 36.04% and 98.49% as shown in table 1.8 and figure 1.8 in this paper which agreed with the result of [**14**] that DABRR improved AWT and ATAT of RR by 41.23% and 30.70% respectively. The little variation could be due to the differences in TQ selection and nature of burst times generated. The same table 1.8 and figure 1.8 showed that RMRR improved AWT, ATAT and NCS of RR by 17. 04%, 12.75% and 97.65% respectively confirmed by the research of [11] which when equation v, vi and vii are applied on the data in their table 2 indicates 21.93%, 16.99% and 66.67% performance improvement on RR with respect to AWT,ATAT and NCS respectively. The difference of the 4% could be attributed to the nature of processes generation and range in their burst times. Still the table 1.8 and figure 1.8 showed that DTSRR improves AWT, ATAT and NCS performance of RR by 7.06%, 7.04% and 86.365 respectively conforming with the result of [15] when equation v, vi and vii are applied to their table 2 which indicates 9.02 and 6.70% their table 4 indicates 12.84% and 9.85% while their table 6 indicates 8.14% and 4.08% performance improvement on AWT and ATAT respectively and respectively as contained in their research.

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

The little variation in the result of this paper and those of the reviewed counts on number of processes and their sizes as can be seen that RMRR used 7 processes with highest burst time 58 in [11], DTSRR separately used 5, 5 and 4 processes with highest burst time 77, 74 and 85 respectively in [15] while this paper used 1000 processes with the highest burst of 360,000.

This then showed that DABRR tops RMRR and DTSRR in the performance ladder in addition to the list it outperformed in [14] namely DQRRR, IRRVQ, SARR, RP-5, MRR and ORR that DTSRR tops in [15]. RMRR is second in the ladder toping just DTSRR and ORR that the DTSRR tops in [15]. Consequently, DTSRR is the third topping just ORR it outperformed in [15]

## 8. SUMMARY OF FINDINGS

The researcher discovered that;

( a). RR will perform better than the improved algorithms under review when TQ is greater than the average of burst times. However discourage since large TQ forces RR to FCFS **[20]**

(b ). The RR outperforms DTSRR if the generated burst time range is large but not when the generated burst times are closed.

(c ). That not all improved newly proposed algorithms are more optimal compared to those proposed before them. This is clearly showed in the case of DTSRR and RMRR of 2016 and DABRR of 2015.

(d ) Most researchers do test their proposed algorithms with few set of data and small size of burst times and then conclude.

## 9. CONCLUSION

The researcher hereby concluded that the proposers of new algorithms should always compare several improved and proposed existing algorithms before proposing. They should always compare their proposed algorithms with other latest proposed algorithms otherwise improvement cannot be improvement. The proposed algorithm be tested many set of data in hundreds and with a large burst times sizes in thousands before conclusion. Thus, by comparing most latest, the RR improved proposed algorithms can form a ladder in which the most optimal can be ascertained for real implementation in operating systems and for optimal performance.

## 10. REFERENCES

[1] L. Datta, “Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice,” *International Journal of Education and Management Engineering*, vol. 02, no. 02, pp 10-19, June 2015, DOI:105815/ijeme.2015.02.02

[2] N. Goel and B.R. Garg, “A Comparative Study of PU Scheduling Algorithms,” *International journal of graphics and Image Processing*, vol. 02, issue 04, pp 245-25, November 2012

[3] A. R. Dash, S. K. Sahu, S. K. Samantra and S. Sabat, “Characteristic Specific Prioritized Dynamic Average Burst Round Robin Scheduling for Uniprocessor and Multiprocessor Environment”, *International Journal of Computer Science, Engineering and Applications*(IJCSEA), Vol.05, no. 4/5, pp 1-20, October 2015, DOI : 10.5121/ijcsea.2015.5501

[4] N. Goel and B. R. Garg, “Improvised Optimum Multilevel Dynamic Round Robin Algorithm for Optimizing CPU Scheduling,” *International Journal of Computer Applications*, Vol. 123, no. 16, pp 1-5, August 2015, ISSN 0975-8887

[5] H. S. Behera, R. Mohanty, S. Sahu and S. K. Bhoi, “Comparative Performance Analysis of Multi-dynamic Time Quantum Round Robin (MDTQRR) Algorithm With Arrival Time,” *Indian Journal of Computer Science and Engineering* (IJCSE),Vol. 02, no. 02, pp. 262-271, April-May 2011, ISSN 0976-5166.

[6] A. Bhardwaj and R. Singh, “Comparative Study of Scheduling Algorithm in Operating Systems”, *International Journal of Computers and Distributed Systems*, vol. 03, no. 01, April-May 2013, ISSN 2278-5183.

[7] G. S. N. Rao, S. V. N. Srinivasu, N. Srinivasu and N. O. Raju “An Augmented Dynamic Round Robin CPU Scheduling Algorithm” *International Journal of Advanced Information in Arts, Science & Management*, vol. 04, no. 04, pp 25-33, April 2015, ISSN: XXXX-XXXX.

[8] P. Banerjee and L. N. Padhy, “ Comparative Analysis of Maximum Performance Round Robin(MPRR) by Dynamic time Quantum with static Time Quantum”, *International Journal of Advanced Research Computer Science and software Engineering* (IJARCSSE), vol. 04, no. 11, pp 372-377, November 2014, ISSN 2277-128X.

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

[9] N. Srinivasu, A. S. V Balakrishna and R. D. Lakshmi, "An Augmented Dynamic Round Robin CPU Scheduling Algorithm", *Journal of Theoretical and Applied Information Technology*, vol. 76, no. 01, pp 118-126, June 2015, eISSN 1817-3195/pISSN 1992-8645.

[10] K. M. R. Mahesh, R. B. Renuka, M. Sreenatha and C. K. Niranjan, "An Improved Approach to Minimized context Switching in Round Robin Scheduling algorithm Using Optimization Techniques", *International Journal of Research in Engineering Technology* (IJRET), vol. 03, issue 04, pp 804-808, April 2014, eISSN 2319-1163/pISSN 2321-7308.

[11] S. Kathuria , P. P. Singh and P. T. Prashant, "A Revamped Mean Round Robin (RMRR) CPU Scheduling Algorithm", *International Journal of Innovative Research in Computer and Communication Engineering* (IJIRCCE), vol. 04, issue 04, pp 6684-6691, April 2016, eISSN 2320-9801/pISSN: 2320-9798.

[12] C. McGuire and J. Lee, "Comparison of Improved Round Robin Algorithms" *Proceedings of the World Congress on Engineering and Computer Science San Francisco*, USA, vol I , pp 22-24, October 2014, eISSN: 2078-0966/pISSN: 2078-0958.

[13] R. Joshi and S. B. Tyagi, "Smart Optimized Round Robin (SORR) CPU Scheduling Algorithm", *International Journal of Advanced Research in computer Science and software Engineering*, vol. 05, issue 07, pp 568-574, July 2015, ISSN: 2277 128X

[14] A. R. Dash, S. K. Sahu and S. K. Samantra " An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum", I*nternational Journal of Computer Science, Engineering and Information Technology* (IJCSEIT), vol. 05, no. 01, February 2015, pp 7-26, Doi:10512/ijcseit.2015.5102

`[15] A Muraleedharan, N. Antony and R. Nandakumar, "Dynamic Time Slice Round Robin Scheduling Algorithm with Unknown Burst Time", *Indian Journal of Science and Technology* Vol.9(8), February 2016, pp 1-6, eISSN 0974-5645/ pISSN:0974-6846, DOI:10.17485/ijst/2016/v9i8/76368

[16] P. Banerjee, T. Zabin, S. Kumai and P. Kumari, "Comparative Performance Analysis of Best Performance Round Robin Scheduling Algorithm (BPRR) Using Dynamic Time Quantum with Priority Based Round Robin (PBRR) CPU Scheduling Algorithm in Real Time System", *International Journal of Electronics and Computer Science Engineering* (IJECSE), vol. 04, no. 02, pp 151-159, 2012, ISSN: 2277-1956

[17] P. T. Sonagara and V. D. Sanghvi, "A Comparative Parameters Analysis of Different Round Robin Scheduling Algorithm Using Dynamic Time Quantum", *International Journal of Science and Research* (IJSR), vol. 03, no. 12, pp 2025-2047, 2012, ISSN: 2319-7064

[18] P. Singh, V. Singh and A. Pandy, "Analysis and Comparison of CPU Scheduling Algorithms", *International Journal of Emerging Technology and Advanced Engineering*, vol. 04, issue 01, pp. 91-95, January 2014.

[19] S. Negi and P. Kalra, "A comparative Performance Analysis of Various Variants of Round Robin Scheduling Algorithm" *International Journal of information and computation Technology*, vol. 04, no. 07, pp 765-772, 2014, ISSN: 0974-2239

[20] S. R. Chavan and P. C.Tikekar "An improved Multilevel Dynamic Round Robin Scheduling Algorithm", *International Journal of Scientific and Engineering Research* (ijser), vol. 04, issue 12, pp. 298-301, December 2013, ISSN:2229-5518

International Journal of Scientific & Engineering Research, Volume 8, Issue 8, August-2017
ISSN 2229-5518

K I Musa PhD
Department of Information Technology
Department of Management and Information Technology,
Abubakar Tafawa Balewa University, Bauchi, Nigeria,
imkabir@atbu.edu.ng
K E. Lasisi PhD
Department of Mathematics,
Abubakar Tafawa Balewa University
Bauchi, Nigeria
kazeemlasisi@gmail.com
J. A Gokir
Currently pursuing masters degree in Computer Science,
Department of Mathematics
Abubakar Tafawa Balewa University
Bauchi, Nigeria.
Email: jgokir@rocketmail.com