Round Robin and Shortest Job First Hybrid Scheduling Algorithm Efficiency Analysis Using Frequency Count and Memory Requirement

Renz Rallion T. Gomez
Author/Student
Isabel Subd., San Nicolas Pob.,
Concepcion, Tarlac, Philippines 2316
decemberavis19@gmail.com

Christopher M. Bermudez
Author/Student
Pias, Camp 7,
Baguio City, Philippines 2600
tupz0799@gmail.com

Vily Kaylle G. Cachero
Author/Student
Poblacion A. Tayug,
Pangasinan, Philippines 2445
cacherokaylle@gmail.com

Eugene G. Rabang
Author/Student
Poblacion Zone-V, Villasis,
Pangasinan, Philippines 2427
raterkracks@gmail.com

ABSTRACT

CPU scheduling algorithms are used to provide a method in queueing and executing instructions. These can be improved by either modifying, combining or creating a new algorithm. This paper aims to determine the improvement of the efficiency of End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis from its predecessors, Shortest Job First (SJF) and Round Robin (RR). The researchers simtulated different test cases wherein various instruction sets are defined to get results that would identify their time and space efficiencies in the form of frequency count and memory requirement, respectively. In the findings across all the test cases, E-EDRR shows scores that are fractions of the other two algorithms' scores. Time efficiency was improved by 42% against the original SJF algorithm and 92% against the RR. Space efficiency was improved by 52% against the original SJF algorithm and 100% against the RR with integrated quicksort algorithm. Based on these findings, the researchers conclude that this hybrid algorithm End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis is successful in being a more efficient scheduling algorithm than its predecessors.

Keywords

Scheduling Algorithm; Round Robin, Shortest Job First; Efficiency; Frequency Count; Memory Requirement;

1. INTRODUCTION

Scheduling Algorithms in Operating Systems provide an established method of queuing instructions for the Central Processing Unit (CPU) to execute. They define unique systematic ways that enables the CPU to execute instructions in efficient ways. Researchers around the globe are trying to develop or improve algorithms to further optimize the processes involved in scheduling tasks/instructions. These improvements are always crucial in making scheduling algorithms as efficient as possible to save resources. These resources include time and memory space. Being able to be efficient in the usage of these resources improves the overall performance of the systems that uses these scheduling algorithms. Our current technology can only rely on these methods of improvement to further revolutionize its capabilities in terms of performance.

The studies regarding efficiency comparisons, especially in CPU scheduling algorithms are somewhat uncommon. In respect to the academic community, these studies are still relevant in determining the appropriate knowledge and information to continue improving algorithms. Some studies specify in improving the metrics that affects the performance of the scheduling. This includes CPU utilization, throughput, turnaround time, waiting time, response time and context switches. Others use simulation or mathematical analysis to calculate efficiencies. This paper uses simulation and observation to analyze time and space efficiencies of the subjected scheduling algorithms.

Popular scheduling algorithms include First Come First Serve (FCFS), Shortest Job First, Round Robin, Priority Algorithm and more. Although these existing algorithms are sufficient, they have their own downsides. For example, Round Robin (RR) has a characteristic that its performance is heavily reliant on its time quantum (TQ) and can become highly inefficient if the TQ is too short causing it to have an abundance of context switches. Another example is the Shortest Job First's (SJF) flaw wherein a congestion occurs on the longer instructions due to SJF's characteristic of prioritizing the shorter instructions first. These issues introduce inefficiencies because of the lack of proper process management.

This paper aims to determine the order of growth of End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis, Shortest Job First and Round Robin. It also aims to determine whether there is a difference in the determined orders of growth. If there are differences, this paper aims to determine whether the hybrid algorithm, E-EDRR, has better efficiency than its predecessors, SJF and RR.

The following research will be in the form of simulation of queuing and executing various instruction sets. The analysis will calculate for the number of times a statement has been executed and the number of Bytes in the memory space used by the algorithms. These simulations are represented as test cases. These test cases include variables of the number of instructions and their length. After said test cases were conducted, a discussion will explain the results acquired. These results will be validated and concluded. Recommendations will also be provided for future references.

2. REVIEW OF RELATED LITERATURE

This chapter presents the related literature and studies both local and foreign sources. This also includes theoretical framework terms, defined conceptually and operationally for clarity.

This paper aims to determine the improvement of SJF and RR through the efficiency analysis of time and space used by E-EDRR but the most common metrics used to further improve scheduling algorithms include:

Turnaround Time: The time required to complete a process (wall clock time). It starts from submission time to completion.

Waiting Time: The time that a process spends in the queue before being executed.

Response Time: The time it takes to respond to an issuance of a command.

Context Switch: The process of switching tasks/thread, given that the current process is saved so it can be continued later on.

Improved CPU scheduling algorithms are usually in the form of combined existing scheduling algorithms but making some simple to complex changes in the algorithms structure. Some of the most popular existing algorithms follows:

1. **The First Come First Serve (FCFS)** or also known as **First In First Out (FIFO)** uses the queue method of scheduling wherein the first to arrive is the first to leave. In reality, it is represented by a line or a lining system.
2. **Shortest Job First (SJF)** analyzes the tasks and executes the shortest one first.
3. **Round Robin (RR)** consecutively executes each task equally given a TQ.
4. **Priority Algorithm** orders processes based on their priority number that is given to each process (given priority number 0 as the highest priority, 1, 2, …, n as lowest priority).
5. **Best Job First (BJF)** queues tasks based on the tasks Priority, Arrival Time and Burst Time (given factor f where: f = Priority + Arrival Time + Burst Time to determine tasks location in the ready queue).

Other studies used both qualitative and quantitative research methods. Researchers evaluate the algorithms given to them in terms of their run-time efficiency [1]. In this paper, the researchers only used quantitative method where we used our findings and comparing them. Other studies focused on the evaluation of the efficiency of predictive schedules using criteria: makespan, total tardiness, flow time, idle time. In terms of efficiency of reactive schedules, these are evaluated using: solution robustness criterion and quality robustness criterion. [2] To simplify, in this paper, the researchers used time and memory space as criteria in determining efficiencies.

Improved Mean Round Robin with Shortest Job First Scheduling (IMRRSJF) [3]: This algorithm has combined features of both RR and SJF, ie, the processes are arranged according to the burst time values and time quantum is calculated as the square root of the product of mean and highest burst time. Whenever a new process comes in, the processes are again sorted in ascending order and time quantum is calculated again. E-EDRR has the same property but the time quantum is calculated based on the current shortest instructions' burst time.

Augmented Dynamic Round Robin scheduling (ADRR) [4]: In this algorithm, processes are executed according to their arrival times. Once a process is executed for the defined time quantum, instead of switching to the next process, it checks whether the resultant burst time of the current process is less than or equal to time quantum value. E-EDRR has a different approach where it checks the longest instruction for the next process.

An experiment done on the Fittest Job First Dynamic Round Robin (FJFDRR) scheduling algorithm submitted a depleting number of context switches, average turnaround time, and average waiting time. [5]. The researchers also conducted experiments on the E-EDRR and it showed lower turnaround time, waiting time and context switches than the RR and SJF.

Time Quantum Based Improved Scheduling Algorithm (TABISA) [6]: This algorithm made use of a Median based time quantum based scheduling algorithm which is a combination of SJF & RR where all the jobs in the queue are first aligned as per their burst time in ascending order and them Round robin is applied for improving the performance. This introduced a median based time quantum as an instruction set wherein E-EDRR introduced an individual analysis of instructions in calculating the time quantum.

Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis (DQRR) [7]: This algorithm first arranges processes in increasing order of burst times, calculates time quantum as the median of the burst-times. After each process execution, the processes are re-arranged such that process with least remaining burst-time will come first, then the process with the highest remaining burst-time comes followed by the process with the second least remaining burst-time and so on. E-EDRR has the same properties but it is the same as TABISA [6] which uses a median as a basis to the time quantum wherein E-EDRR the shortest instruction's burst time as time quantum.

Enhanced Precedence Scheduling Algorithm with Dynamic Time (EPSADTQ) [8]: This algorithm used the typical round robin but introduced priority assignment to sort the instructions. E-EDRR did not make use of priority queues but instead used the SJF method to queue the instructions.

Hybrid Scheduling and Dual Queue Scheduling [9]: This algorithm was an optimization tool. It used two queues, the waiting and the execution queues. E-EDRR also used this dual queue strategy to store the instructions. This resulted in a more organized separation and sorting of instructions.

A comparative study of dispatching rules in dynamic flowshops and jobshops [10]: This showed the results of mathematical analysis using simulation that differentiated scheduling algorithms in the form of statistical functions. This paper also made use of simulation but the researchers calculated for the time and space efficiencies using frequency count and memory requirement.

3. METHODS

The purpose of this chapter is to explain methods and the options upon achieving the results of the research. The researcher's had decided to evaluate two types of algorithm efficiencies in which are the time and space efficiencies.

Time Efficiency was chosen by the researchers due to the fact that it is a quick method to collect time intervals that measures the amount of time that the algorithm or instructions is executed. This is used to test the efficiency of the E-EDRR algorithm's execution time that allows the researchers to compare and conclude the results to other algorithms in terms of its efficiency of its execution time. Space efficiency is a measurement of the amount of memory that is needed for an algorithm to be executed.

In addition, space efficiency is the ability to store and manage data that can consume the least amount of spaces without any collision on the performance of the algorithm, which will result in a less memory requirement. This is used to test the efficiency of the E-EDRR algorithm's memory allocation that allows the researcher to compare and conclude the results to other algorithms in terms of its efficiency of its space or memory.

These algorithm efficiencies will be integrated among the researcher's algorithm since both of these efficiencies are reliable for the research. A quantitative arrangement is specifically applicable for the purpose of this research, where the connection between several variables has to be interpreted through testing and simulation of test cases.

3.1 E-EDRR

The End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis functions as a method of queuing tasks that the CPU will process. E-EDRR is an improved Round Robin that uses the Shortest Job First analysis to compare tasks and the end to end method to execute different tasks.

It aims to reduce three metrics, the first is the time it takes to complete a task, the second is the time it takes for the incoming newly arrived task that process the tasks to be executed, and lastly is the number of times that CPU will switch between tasks.

3.1.1 Assumptions

1. Burst times are known.
2. A batch of tasks or an individual task is received and the algorithm is applied.

3.1.2 Interpretation

1. Ready queue (Q1) is the queue that holds the tasks that are ready for execution.
2. Tasks in Q1 are sorted based on their burst time.
3. Time quantum (TQ) is calculated by:
   TQ = current shortest task's burst time ----- [*equation 2*]
4. Algorithm is applied on the Q1 and is reapplied until Q1 is empty.
5. Shortest and longest tasks are executed consecutively.
6. If upon execution, the longest task is incomplete, the longest task's progress is saved and the burst time is reduced by TQ.
7. Completed tasks are removed from Q1.
8. Newly arrived tasks are added to the Q1 and updated.

3.1.3 Pseudocode of the E-EDRR

```
Let TQ be the time quantum.
Let NA be the newly arrived processes.
Let Q1 be the ready queue
1. if(NA == true)
     {enqueue NA to Q1,
     repeat step 1}
     else
       {proceed to step 2}
2. if(Q1 != empty)
     {sort tasks according to burst time,
     proceed to step 3}
   else
       {proceed to step 1}
3. Determine the TQ by using [equation 2]
4. if(Q1.length != 1)
     {execute shortest task,
     execute longest task}
   else
     {execute shortest task}
5. if(longest task != complete)
     {Longest task's progress is saved and its burst time is
     reduced by TQ}
   else
     {proceed to step 6}
6. Dequeue completed tasks from Q1 and proceed to step
   1
```

3.2 Shortest Job First

The Shortest Job First (SJF) scheduling algorithm processes the smallest execution time that needs to be executed next, this reduces the average waiting time to other processes that is waiting for execution. The researchers used this as a tool to compare the E-EDRR algorithm as it's competitor in order to test the full capabilities and efficiency of the algorithm.

3.2.1 Assumptions

1. Burst times are known.
2. A batch of tasks or an individual task is received and the algorithm is applied.

3.2.3 Interpretation

1. Ready queue (Q1) is the queue that holds the tasks that are ready for execution.
2. Tasks in Q1 are sorted based on their burst time.
3. Time Quantum (TQ) is calculated by:
   TQ = current shortest task's burst time ----- [*equation 2*]
4. Algorithm is applied on the Q1 and is reapplied until Q1 is empty.
5. Completed tasks are removed from Q1.
6. Newly arrived tasks are added to the Q1 and is updated.

3.2.3 Pseudo Code of the SJF

```
Let BT be the burst time
Let NA be the newly arrived task
Let Q1 be the ready queue
1. if(NA == true) {
       enqueue NA to Q1;
```

```
      repeat step 1;
   } else { proceed to step 2 }
2. sort task according to BT;
   proceed to step 3;
3. if (Q1 != empty) {
     execute shortest task;
     repeat step 3;
   } else { proceed to step 4}
4. Dequeue completed tasks from Q1 and proceed to
   step 1
```

3.3 Round Robin

The Round Robin (RR) scheduling algorithm processes a fix time quantum in order to execute the task, each processes are executed for a given period of time. The researchers used this as a tool to compare the E-EDRR algorithm as its competitor in order to test the full capabilities and efficiency of the algorithm. However, the researchers decided to include a sorting mechanism to the algorithm, specifically the quick sort, to sort the instructions as it levels the complexity of the other two scheduling algorithms.

3.3.1 Assumptions

1. Time quantum is fixed to 25.
2. Burst times are known.
3. A batch of tasks or an individual task is received and the algorithm is applied.

3.3.2 Interpretation

1. Ready queue (Q1) is the queue that holds the tasks that are ready for execution.
2. Tasks in Q1 are sorted based on their burst time.
3. Time Quantum is fixed given by the user
4. Algorithm is applied on the Q1 and is reapplied until Q1 is empty.
5. If upon execution, the longest task is incomplete, the longest task's progress is saved and the burst time is reduced by TQ.
6. Completed tasks are removed from Q1.

3.3.3 Pseudo Code for RR

```
Let TQ be the time quantum
Let NA be the newly arrived processes
Let Q1 be the ready queue
Let BT be the burst time of each task
1. if(NA == true) {
      enqueue NA to Q1;
      repeat step 1;
   } else {proceed to step 2}
2. get BT of  NA
   proceed to step 3;
3. sort tasks according to BT
   proceed to step 4;
4. if (BT < TQ) {
      execute task until completed;
   } else {
      execute task according to TQ;
   }
   proceed to step 5;
5. If (Q1 != empty) {
        repeat step 3;
   } proceed to step 6;
6. Dequeue completed tasks from Q1 and proceed to
   step 1
```

4. FINDINGS

All test cases were performed with the consideration of the following assumptions:

1. Processes are executed in a single processor.
2. Processes are CPU bound.
3. Number of processes and BTs are initially known.
4. SJF and RR are used as benchmarking algorithms.
5. RR will have a TQ of 25 in respect to the test cases' average BT.

The metrics used in determining the results are variable statement_counter that counts the number of statements executed indicated in the TE column to verify the time efficiency and variable space_counter that counts the number of Bytes used by the algorithm indicated in the SE column to verify the space efficiency.

Test Case 1: We assumed five (5) processes wherein they have equal BTs (as shown in Table 2 below).

[Table 2: Test Case 1]

Task	Burst Time
T0	25
T1	25
T2	25
T3	25
T4	25

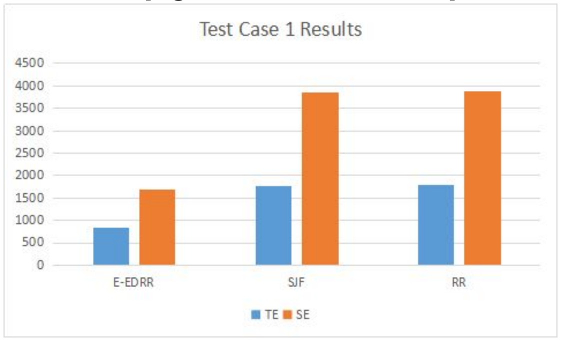
Table 3 shows the comparative results of E-EDRR against the benchmarking algorithms.

[Table 3: Test Case 1 Results]

Algorithm	TE	SE
E-EDRR	847	1686
SJF	1768	3866
RR	1788	3872

In the Table 3 above, E-EDRR gave time efficiency results that are approximately 50% of the SJF and RR.

[Figure 1: Test Case 1 Results]



Test Case 2: We assumed five (5) processes wherein they have increasing BTs (as shown in Table 4 below).

[Table 4: Test Case 2]

Task	Burst Time
T0	19
T1	22
T2	25
T3	28
T4	31

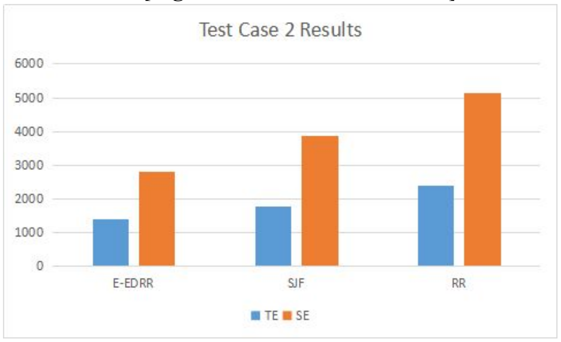
Table 5 shows the comparative results of E-EDRR against the benchmarking algorithms.

[Table 5: Test Case 2 Results]

Algorithm	TE	SE
E-EDRR	1391	2814
SJF	1768	3866
RR	2392	5156

In the Table 5 above, E-EDRR shows results that are 70% of the SJF's results both in time and space efficiencies. On the other hand, the results show only about 60% of the RR's time efficiency results and 55% of the RR's space efficiency result. Still indicating that the E-EDRR is more efficient than the other algorithms.

[Figure 2: Test Case 2 Results]



Test Case 3: We assumed five (5) processes wherein they have decreasing BTs (as shown in Table 6 below).

[Table 6: Test Case 3]

Task	Burst Time
T0	31
T1	28
T2	25
T3	22
T4	19

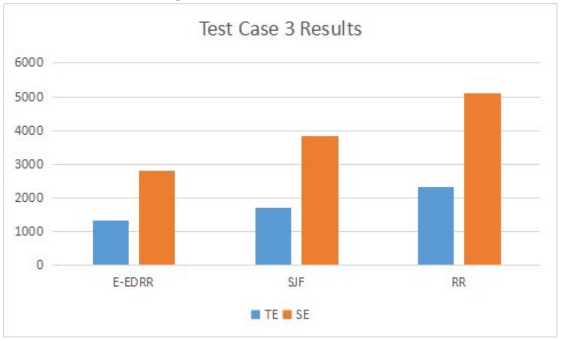
Table 7 shows the comparative results of E-EDRR against the benchmarking algorithms.

[Table 7: Test Case 3 Results]

Algorithm	TE	SE
E-EDRR	1311	2814
SJF	1688	3826
RR	2312	5116

In the Table 7 above, E-EDRR shows results that are approximately 75% of the SJF's and are only about 55% of the RR's in both time and space efficiencies. Still indicating that the E-EDRR is more efficient than the other algorithms.

[Figure 3: Test Case 3 Results]



Test Case 4: We assumed five (5) processes wherein they have random BTs (as shown in Table 8 below).

[Table 8: Test Case 4]

Task	Burst Time
T0	27
T1	16
T2	35
T3	26
T4	40

Table 9 shows the comparative results of E-EDRR against the benchmarking algorithms.

[Table 9: Test Case 4 Results]

Algorithm	TE	SE
E-EDRR	1349	2762
SJF	1726	3802
RR	2937	6376

In the Table 9 above, E-EDRR shows time efficiency results that is about 78% of the SJF's time efficiency result and 72% of the SJF's space efficiency. In relation to RR, the results are only about 46% in time efficiency and about 43% in space efficiency.

[Figure 4: Test Case 4 Results]

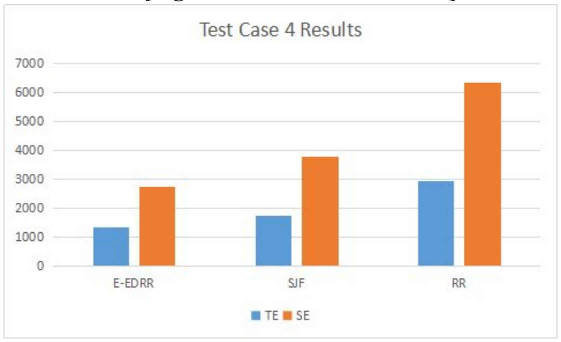


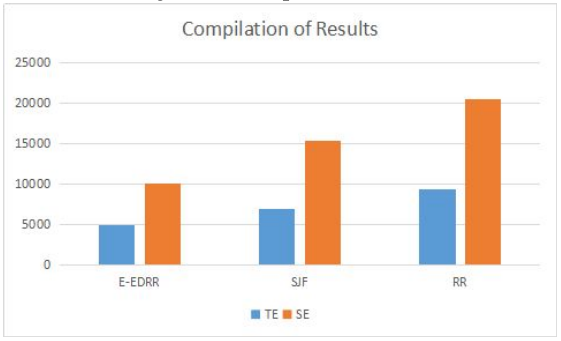
Table 10 below shows the compilation of the results of the test cases.

[Table 10: Compilation of Results]

Algorithm	TE	SE
E-EDRR	4898	10076
SJF	6950	15360
RR	9429	20520

In the Table 10 above, the compilation of results shows significant lead of E-EDRR in terms of time efficiency by 40% to the SJF and 93% to the RR. While in terms of space efficiency, E-EDRR shows a significant lead of 52% to the SJF and 100% lead on the RR.

[Figure 5: Compilation of Results]



4.1 Discussions

Across all test cases, the E-EDRR has shown a significant lead by getting lower scores in both time and space efficiencies ranging from 40% to 100%.

In the test case 1, the algorithms has shown results wherein the instructions have equal lengths. The E-EDRR has shown that it is approximately 100% more efficient than the other two algorithms, RR and SJF. These two algorithms had about the same scores because their execution was about similar in the given test case.
In the test case 2 and 3, the E-EDRR remains undefeated by 27% and 37% lead against the second rank, SJF, in time and space efficiencies, respectively. E-EDRR also got a 72% and 83% lead against the third rank, RR, in time and space efficiencies, respectively. SJF was significantly better than RR by about 34% in both time and space efficiencies. This shows that in instructions where the length is increasing or decreasing, the SJF's execution technique has an advantage because it retains the number of context switches unlike RR.
In the test case 4, the results were almost similar to the previous two test cases except the RR had significant increase in space efficiency score.

5. CONCLUSIONS

In the findings, the results have shown improvement of the original algorithms by combining them into a single hybrid algorithm, specifically, as tested, the End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis. Time efficiency was improved by 42% against the original SJF algorithm and 92% against the RR with integrated quicksort algorithm. Space efficiency was improved by 52% against the original SJF algorithm and 100% against the RR with integrated quicksort algorithm.
As observed from the findings, the efficiency of the E-EDRR varies based on the type of instruction set to be executed. From the findings, if the instruction set consists of instructions with equal burst times, the efficiency is as its best. The researchers also found that if the instructions have varying burst times, even if they are randomized or not, they have about the same efficiencies that are worse than if they have equal burst times.
As observed from the findings, the orders of growth of E-EDRR in both time and space efficiencies are n^4 where n is the number of instructions. In terms of time efficiency, the order of growth of the E-EDRR is the same as the two other algorithms. In terms of space efficiency, the order of growth of the E-EDRR is lower than the other two algorithms since their order of growth is n^5. Like E-EDRR, RR its space efficiency depends on the type of instruction set. Their difference allows E-EDRR to have better efficiency in a certain degree based on that difference itself.
Therefore, the researchers conclude that this hybrid algorithm End to End Dynamic Round Robin (E-EDRR) Scheduling Algorithm Utilizing Shortest Job First Analysis is successful in being a more efficient scheduling algorithm than its predecessors.

6. RECOMMENDATIONS

For future references, the researchers recommend taking a different approach in testing these algorithms where in the coding of the algorithm does not require recursive methods to lessen the variables of efficiency calculation.
Since the researchers have used actual codes to simulate and test the algorithms because it the most accurate approach in testing these algorithms. To generalize these analysis, the more suitable approach would be through mathematical analysis of the algorithms.

7. REFERENCES

[1] Shah, S. N. M., Mahmood, A. K. Bin, & Oxley, A. (2009) Hybrid Scheduling and Dual Queue Scheduling *2009 2nd IEEE International Conference on Computer Science and Information Technology*

[2] Singh, A., Goyal, P., Batra, S. An Optimized Round Robin Scheduling Algorithm for CPU Scheduling *International Journal on Computer Science and Engineering Vol. 02, 2010*

[3] Radhe Shyam and Sunil Kumar Nandal 2014 Improved Mean Round Robin with Shortest Job First Scheduling *International Journal of Advanced Research in Computer Science and Software Engineering, Vo.4, Issue.7, pp. 170-177*

[4] N Srinivasu, A.S.V Balakrishna and R.Durgalakshmi 2015 An Augmented Dynamic Round Robin CPU Scheduling Algorithm *Journal Of Theoretical and Applied Information Technology, Vol.76, No 1, pp . 119-124.*

[5] Matarneh R 2009 Self-Adjustment time quantum in round robin algorithm depending on burst time of the now running processes *American J. of Applied Sci. Vol 6 No. 10 pp 1831-7*

[6] Lalit Kishor and Dinesh Goyal 2013 Time Quantum Based Improved Scheduled Algorithm *International Journal of Advanced Research in Computer Science and Software Engineering, Vol .3, Issue. 4,pp. 955-962.*

[7] H.S.Behera,R.Mohanty and Debashree Nayak 2010 A New Proposed Dynamic Quantum with ReAdjusted Round Robin Scheduling Algorithm and its Performance Analysis *International Journal of Computer Applications, Vol. 5-No.5, pp. 10-14.*

[8] G.Siva Nageswara Rao, S.V.N. Srinivasu, N. Srinivasu and G. Ramakoteswara Rao 2015 Enhanced Precedence Scheduling Algorithm with Dynamic Time Quantum (EPSADTQ) *Research Journal of Applied Sciences, Engineering and Technology vol 10 No.8: 938-941 ISSN: 2040- 7459,pp.931-941*

[9] Shah, S. N. M., Mahmood, A. K. Bin, & Oxley, A. (2009) Hybrid Scheduling and Dual Queue Scheduling *2009 2nd IEEE International Conference on Computer Science and Information Technology*

[10] Rajendran C, Holthaus O. 1996 A comparative study of dispatching rules in dynamic flowshops and jobshops *European Journal of Operational Research. 1999*