Comparisons of Improved Round Robin Algorithms

Christopher McGuire and Jeonghwa Lee

***Abstract*—Many altered versions of the Round Robin CPU scheduling algorithm have been created to fix the shortcomings of the Standard Round Robin algorithm. When these improved Round Robin algorithms are first created, they are always compared to the Standard Round Robin algorithm to ensure that they provide an improvement over it. Occasionally, they are compared to one other improved Round Robin algorithm for comparison as well, however little work has been done to compare several of these algorithms simultaneously to observe which ones make the greatest improvements over the Standard Round Robin algorithm. This research seeks to answer this query by comparing five improved Round Robin algorithms.**

***Keywords*—Context Switch, Round Robin, Process, Scheduling Algorithm, Turnaround Time, Waiting Time.**

I. Introduction

THE Round Robin (RR) CPU scheduling algorithm, referred to hereafter as Standard RR, is commonly used in time sharing and real time operating systems [1], [2], [3], [8], [17] because it keeps response time low [1] and gives each process a fair share of time to use the CPU. Despite these advantages, it is well known that the Standard RR algorithm suffers from several disadvantages; those being low throughput, high turnaround time, high waiting time, and a high amount of context switches [1], [8], [15]. Other researchers have proposed improved RR algorithms to minimize these shortcomings. These algorithms have been compared to the Standard RR algorithm to prove that they produce better results than it, and occasionally they are compared to one or two other improved RR algorithms. Rarely have larger numbers of improved RR algorithms been compared at the same time to see how each of these improved RR algorithms compare to the others.

This paper compares the effectiveness of five improved RR algorithms; A New RR (AN RR) [1], Optimized RR [2], Priority-Based RR [8], Adaptive RR [15], and Efficient RR [17]. The effectiveness of the Standard RR algorithm was measured to use as a base-line for these improved RR algorithms.

Section II describes each of the RR algorithms. Section III explains the experiment used to analyze the effectiveness of each algorithm. Section IV summarizes the results, and Section V discusses the results.

II. Descriptions of the Round Robin Algorithms

The performance of Standard RR is greatly affected by the choice of the time quantum [15], [17]. If the time quantum is very small, then the RR algorithm degenerates into a Processor Sharing algorithm, and the overhead due to frequent context switching becomes costly. If the time quantum is too large, then the RR algorithm degenerates into a First-Come-First-Served (FCFS) algorithm [3], [4], [15].

A. AN RR

AN RR focuses on calculating an ideal time quantum [1]. It is like Standard RR with exception that each time a process moves in or out of the ready queue, the time quantum is recalculated. If the ready queue is empty, then the time quantum equals the burst time of the running process. Otherwise, the time quantum equals the average burst time of the processes in the ready queue.

B. Optimized RR

Optimized RR is like Standard RR with the following exceptions. Optimized RR consists of two phases. During phase 1, processes are executed in order just like they are in Standard RR, and each process runs for one time slice. During phase 2, the time quantum is doubled, and processes are executed in the order of their remaining burst times with shorter times running before longer times. After each process has run for one time slice, the phase shifts back to phase one [2]. In [2], no information was given as to what would happen if a process arrived mid-phase. This paper assumes that processes that arrive mid-phase will not get a chance to run until the next phase. This paper also assumes the time quantum resets to its initial value after the second phase.

C. Priority-Based RR

Priority-Based RR combines elements of Priority scheduling and RR scheduling. Priority-Based RR consists of two phases. During phase 1, processes are executed in RR fashion in the order of their default priorities, and each process runs for one time slice [8]. During phase 2, processes are assigned new priorities based on their remaining burst times with shorter remaining burst times receiving higher priorities. Processes are executed in the order of their new priorities and each process runs to completion [8]. This paper assumes that processes that arrive mid-phase will not get a chance to run until the next phase.

D. Adaptive RR

Like AN RR, Adaptive RR focuses on calculating an ideal time quantum [15]. Adaptive RR is like the Standard RR algorithm with the following exceptions. First, processes are sorted by their burst times with the shorter processes at the front of the ready queue. Next, the adaptive time quantum is calculated. If the number of processes in the ready queue is even, then the time quantum equals the average burst time of all the processes. Otherwise, the time quantum equals the burst time of the process in the middle of the ready queue. Any processes that arrive in the middle of the algorithms execution, are added at the end of the queue and do not run during the current round. After each of the initial processes have had a chance to run, the process repeats.

Manuscript received August 6, 2014; revised August 6, 2014. This work was supported in part by Shippensburg University under the Student/Faculty Research Engagement (SFRE) Grant.

Christopher McGuire is with the Department of Computer Science and Engineering, Shippensburg University, Shippensburg, PA 17257. (e-mail: cm7602@cs.ship.edu).

Jeonghwa Lee is with the Department of Computer Science and Engineering, Shippensburg University, Shippensburg, PA 17257. To whom correspondence should be addressed. http://www.cs.ship.edu/~jlee (e-mail: jlee@ship.edu).

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

E. Efficient RR

Efficient RR combines elements of the Shortest Remaining Time (SRT) algorithm and the Standard RR algorithm. In the SRT algorithm, the process with the shortest remaining burst time is always selected to run, and preemption can occur whenever a new process arrives. One of the downsides to SRT is that processes with long remaining burst times can suffer from starvation [1], [3].

Efficient RR is just like the SRT algorithm, but instead of preemption occurring whenever a new process arrives, preemption only occurs at the end of the time slice. At the end of the time slice, when it comes time to select a process to run, the process with the shortest remaining burst time is always selected [16], [17]. Long processes can suffer from starvation in Efficient RR just like in SRT.

III. Simulation

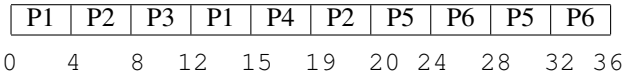
The following simulation gives an example of how each algorithm would schedule a set of processes for a given time quantum (when applicable to the algorithm). For all algorithms assume the following list of processes given in Table 1. In the case where two processes arrive at the same time, the process listed first is assumed to have arrived just slightly before the next process.

TABLE I. Simulated List of Processes

Process	Arrival Time	Burst Time	Priority
P1	0	7	2
P2	0	5	1
P3	3	4	6
P4	5	4	4
P5	10	8	3
P6	13	8	5

A. Standard Round Robin

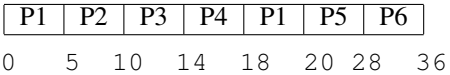
Gantt Chart



Time Quantum = 4
Average Turnaround Time = 17.17
Average Waiting Time = 11.17
Number of Context Switches = 9

B. AN RR

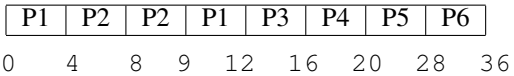
Gantt Chart



Time Quantum calculated by algorithm
Average Turnaround Time = 15.83
Average Waiting Time = 9.83
Number of Context Switches = 6

C. Optimized RR

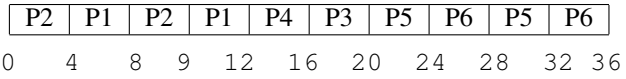
Gantt Chart



Time Quantum = 4 (Phase One), 8 (Phase Two)
Average Turnaround Time = 15.00
Average Waiting Time = 9.00
Number of Context Switches = 7

D. Priority-Based RR

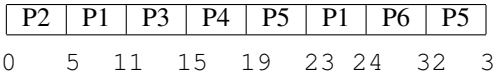
Gantt Chart



Time Quantum = 4
Average Turnaround Time = 15.67
Average Waiting Time = 9.67
Number of Context Switches = 9

E. Adaptive RR

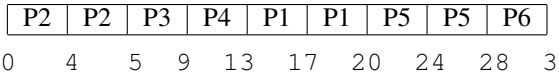
Gantt Chart



Time Quantum calculated by algorithm
Average Turnaround Time = 16.67
Average Waiting Time = 10.67
Number of Context Switches = 7

F. Efficient RR

Gantt Chart



Time Quantum = 4
Average Turnaround Time = 13.33
Average Waiting Time = 7.33
Number of Context Switches = 8

IV. Numberical Experiment

The hardware specs for the machine used to run this experiment were as follows:

- Operating System: Windows 8
- Processor: Intel Core i7-3630QM CPU @ 2.4GHz
- Installed memory (RAM): 8.00 GB (7.89 GB usable)
- System: 64-bit Operating System, x64-based processor

After the program was written, 10 sets of processes were gathered. These sets of processes were selected so that the RR algorithms would face a variety different situations that could occur in a group of processes. For example, some groups had processes that all arrived at time 0, while others had processes that arrived at different times. Some groups had a small number of processes, such as 5, while others had many processes, such as 30 or 10,000. A variety of burst time sizes were used.

Each set of processes was run under each algorithm using four different time quanta. The choice of time quanta for each set of processes was standardized based on the burst times of the processes in the set. The first time quantum was the smallest number that would make the Standard RR algorithm degenerate to the FCFS algorithm for that set of processes. The second, third, and fourth time quanta were three-fourths, one-half, and one-fourth of this number, rounded to the nearest whole number. For example, if 20 was the smallest number that would make Standard RR degenerate to FCFS for the set of processes, then the 4 time quanta would be 5, 10, 15, and 20.

With 10 sets of processes and 4 time quanta per set, this led to 40 different simulations. For each simulation ran, the RR algorithms were ranked according their average turnaround time using the fractional ranking method. The same was done for the average waiting time and the number

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

of context switches. The lowest value was ranked with a 1, the next lowest value with a 2, and so on.

A Friedman statistical test was used to determine if the algorithms differed in their rankings [5], [13]. The null and alternative hypotheses were as follows:

Average Turnaround Time:

- $\mathbf{H}_0$: There are no algorithms which are ranked differently based on their average turnaround time compared to the other algorithms.
- $\mathbf{H}_A$: There is at least one algorithm which is ranked differently based on its average turnaround time compared to at least one other algorithm.

Average Waiting Time:

- $\mathbf{H}_0$: There are no algorithms which are ranked differently based on their average waiting time compared to the other algorithms.
- $\mathbf{H}_A$: There is at least one algorithm which is ranked differently based on its average waiting time compared to at least one other algorithm.

Number of Context Switches:

- $\mathbf{H}_0$: There are no algorithms which are ranked differently based on their number of context switches compared to the other algorithms.
- $\mathbf{H}_A$: There is at least one algorithm which is ranked differently based on its number of context switches compared to at least one other algorithm.

After determining whether the null hypotheses should be accepted or rejected, the sum of the ranks and the mean rank for each algorithm for each category was calculated. The differences of the mean ranks between each algorithm was calculated as well as the critical value for the difference of the mean ranks in order to determine which algorithms' rankings differed.

Table 2 shows the sum of ranks and mean ranks for the RR algorithms for the average turnaround time. For any two algorithms, the closer the sums and closer the averages, the more closely the algorithms ranked. The Chi-squared (χ^2) value yielded by the Friedman test for the average turnaround time was 57.08. The critical χ^2 value at a 95% confidence level for 5 degrees of freedom is 11.07. The differences of the mean ranks and the critical value are shown in Table 3. The critical value for the difference of mean ranks serves as the cutoff line; any difference that is greater than the critical value indicates that the two algorithms ranked differently.

Table 4 shows the sum of ranks and mean ranks for the RR algorithms for the average waiting time. The χ^2 value yielded by the Friedman test for the average waiting time was 53.78. The differences of the mean ranks and the critical value are shown in Table 5.

Table 6 shows the sum of ranks and means rank for the RR algorithms for the number of context switches. The χ^2 value yielded by the Friedman test for the number of context switches was 204.55. The differences of the mean ranks and the critical value are shown in Table 7.

TABLE II. RANK SUMMARY FOR AVERAGE TURNAROUND TIME

RR Algorithm	Sum of Ranks	Mean Rank
Standard	201.5	5.04
AN	148	3.70
Adaptive	137	3.43
Efficient	68.5	1.71
Optimized	139.5	3.49
Priority-Based	142	3.55

TABLE III. DIFFERENCES OF THE MEAN RANKS FOR AVERAGE TURNAROUND TIME

	Standard	AN	Adaptive	Efficient	Optimized
AN	1.338*	x	x	x	x
Adaptive	1.613*	0.275	x	x	x
Efficient	3.325*	1.988*	1.713*	x	x
Optimized	1.550*	0.213	0.063	1.775*	x
Priority-Based	1.488*	0.150	0.125	1.838*	0.063
Critical Value	0.581				

*Value is greater than the critical value

TABLE IV. RANK SUMMARY FOR AVERAGE WAITING TIME

RR Algorithm	Sum of Ranks	Mean Rank
Standard	193.5	4.84
AN	149	3.73
Adaptive	149.5	3.74
Efficient	66	1.65
Optimized	132.7	3.32
Priority-Based	145.5	3.64

V. CONCLUSION REMARKS

A. Average Turnaround Time

Based on the χ^2 result for the average turnaround time, there is enough evidence to reject the null hypothesis for the average turnaround time at the 95% confidence level meaning that there must be at least one algorithm which is rated differently. The results in Table 3 show that Efficient RR and Standard RR ranked differently from the other algorithms. Efficient RR will generally provide a better average turnaround time than the other algorithms, and Standard RR will generally provide a worse average turnaround time than the other algorithms. The evidence suggests that that AN RR, Adaptive RR, Optimized RR, and Priority-Based RR do not perform substantially better or worse than each other for the average turnaround time.

B. Average Waiting Time

Based on the χ^2 result for the average waiting time, there is enough evidence to reject the null hypothesis for the average waiting time at the 95% confidence level meaning that there must be at least one algorithm which is rated differently. The results in Table 5 show that Efficient RR and Standard RR ranked differently from the other algorithms. Efficient RR will generally provide a better average waiting time than the other algorithms, and Standard RR will generally provide a worse average waiting time than the other algorithms. The evidence suggests that that AN RR, Adaptive RR, Optimized RR, and Priority-Based RR do not perform substantially better or worse than each other for the average waiting time.

C. Number of Contact Switches

Based on the χ^2 result for the number of contact switches, there is enough evidence to reject the null hypothesis for the number of contact switches at the 95% confidence level meaning that there must be at least one algorithm which is rated differently. The results in Table 7 show that Standard RR ranked differently from the other algorithms. Standard RR will generally provide a worse number of contact switches than the other algorithms. The evidence suggests that that AN RR, Adaptive RR, Efficient RR, Optimized RR, and Priority-Based RR do not perform substantially better or worse than each other for the number of contact switches.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE V. DIFFERENCES OF THE MEAN RANKS FOR AVERAGE WAITING TIME

	Standard	AN	Adaptive	Efficient	Optimized
AN	1.113*	x	x	x	x
Adaptive	1.100*	0.013	x	x	x
Efficient	3.188*	2.075*	2.088*	x	x
Optimized	1.520*	0.408	0.420	1.668*	x
Priority-Based	1.200*	0.087	0.100	1.988*	0.320
Critical Value	0.609				

*Value is greater than the critical value

TABLE VI. RANK SUMMARY FOR NUMBER OF CONTEXT SWITCHES

RR Algorithm	Sum of Ranks	Mean Rank
Standard	203	5.08
AN	135	3.38
Adaptive	152	3.80
Efficient	143.5	3.59
Optimized	142.5	3.56
Priority-Based	151	3.78

D. Summary

Looking at the ranking for the average turnaround time and the average waiting time, Efficient RR performed better than the other algorithms by a substantial lead. Even though it performed so well, recall that large processes can suffer from starvation under Efficient RR. This is a major drawback to this algorithm because RR algorithms are supposed to keep response time low [1] and give each process a fair share of time to use the CPU. Even though it performed very well for the average turnaround time and average waiting time, it is not an ideal RR algorithm to use due to starvation.

While AN RR, Adaptive RR, Optimized RR, and Priority-Based RR all are substantial improvements over Standard RR, the evidence suggests that none of these four algorithms are substantially better than each other. Any one of these four algorithms would be equally valid choices for CPU scheduling based on their performance relative to each other; it just comes down to which is the most practical for a given situation.

TABLE VII. DIFFERENCES OF THE MEAN RANKS FOR NUMBER OF CONTEXT SWITCHES

	Standard	AN	Adaptive	Efficient	Optimized
AN	1.700*	x	x	x	x
Adaptive	1.275*	0.425	x	x	x
Efficient	1.4875*	0.2125	0.2125	x	x
Optimized	1.513*	0.1.875	0.2375	0.025	x
Priority-Based	1.300*	0.400	0.025	0.188	0.2125
Critical Value	0.823				

*Value is greater than the critical value

REFERENCES

[1] Abbas Noon, Ali Kalakech and Seifedine Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average," *International Journal of Computer Science*, vol. 8, no. 1, May 2011, p. 224-229.

[2] Ajit Singh, Priyanka Goyal and Sahil Batra, "Optimized Round Robin Scheduling Algorithm for CPU Scheduling," *International Journal on Computer Science and Engineering*, vol. 02, no. 07, 2010, p. 2383-2385.

[3] Avi Silberschatz, Peter Baer Galvin, and Greg Gagne, "Operating Systems Concepts Essentials," 7th ed. USA: John Wiley and Sons, Inc., 2011.

[4] Bashir Alam, M. N. Doja, R. Biswas, "Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic," *International Conference on Computer and Electrical Engineering*, 2008.

[5] D. Stricker. "Friedman," *BrightStat.com*, [online] 2014. Accessed: 18 June 2014.

[6] H. S. Behera, Rakesh Mohanty, Sabyasachi Sahu and Sourav Kumar Bhoi, "Comparative Performance Analysis of Multi-dynamic Time Quantum Round Robin (mdtqrr) Algorithm with Arrival Time," *Indian Journal of Computer Science and Engineering*, vol. 2, no. 2, Apr-May 2011.

[7] H. S. Behera, R. Mohanty, and D. Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis," *International Journal of Computer Applications*, vol. 5, no. 5, p. 10-15, Aug. 2010.

[8] Ishwari Singh Rajput and Deepa Gupta, "A Priority based Round Robin CPU Scheduling Algorithm for Real Time System," *International Journal of Innovations in Engineering and Technology*, vol. 1, no. 3, Oct. 2012, p. 1-11.

[9] Pallab Banerjee, Probal Banerjee, and Shaweta Sonali Dhal, "Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum," *International Journal of Innovative Technology and Exploring Engineering*, vol. 1, issue 3, Aug. 2012, p. 56-62.

[10] Prof. Rakesh Mohanty and et al, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm," *Institute of Engineering and Technology*, 2010.

[11] P. Surendra Varma, "A Finest Time Quantum for Improving Shortest Remaining Burst Round Robin (SRBRR) Algorithm," *Journal of Global Research in Computer Science*, vol. 4, no. 3, Mar. 2013, p. 10-15.

[12] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes," American Journal of Applied Sciences, vol 6, no. 10, 2009.

[13] Richard Lowry. "Subchapter 15a. The Friedman Test for 3 or More Correlated Samples," *Concepts & Applications of Inferential Statistics*, [online] 2013. Accessed: 18 June 2014.

[14] Sanjay Kumar Panda and Saurav Kumar Bhoi, "An Effective Round Robin Algorithm using Min-Max Dispersion Measure, *International Journal on Computer Science and Engineering*, vol. 4, no. 01, Jan. 2012.

[15] Saroj Hiranwal and Dr. K.C. Roy, "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice," *International Journal of Computer Science and Communication*, vol. 2, no. 2, July-Dec. 2011, p. 319-323.

[16] Sukumar Babu B., Neelima Priyanka N. and Dr. P. Suresh Varma, "Optimized Round Robin CPU Scheduling Algorithm," *Global Journal of Computer Science and Technology*, vol. 12, Issue 11, 2012, p. 21-25.

[17] Sukumar Babu B., Neelima Priyanka N. and Sunil Kumar B., "Efficient Round Robin CPU Scheduling Algorithm," *International Journal of Engineering Research and Development*, vol. 4, Ino. 9, Nov. 2012, p. 36-42.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)