A COMPARATIVE ANALYSIS OF TIME AND SPACE EFFICIENCY OF TRADITIONAL ROUND ROBIN AND INTELLIGENT ROUND ROBIN SCHEDULING ALGORITHMS

Olivete, Wyli Charles L.
University of the Cordilleras
Governor Pack Road, Baguio City
azeptoblade@gmail.com

Caranto, Hannah S.
University of the Cordilleras
Governor Pack Road, Baguio City
anna_caranto@yahoo.com

Fernandez, Jeune Vincent D.
University of the Cordilleras
Governor Pack Road, Baguio City
mcjeunefernandez@gmail.com

Cabiara, Cecilia Agatha R.
University of the Cordilleras
Governor Pack Road, Baguio City
cabiara.ca@gmail.com

ABSTRACT

Algorithm efficiency and scheduling algorithms can affect overall computer efficiency. Scheduling algorithms can be improved by combining it with other features from other algorithms, and this is mostly done with the Round Robin (RR) algorithm because of its timesharing trait. The intelligent RR algorithm is an improvement of the traditional RR algorithm with better system performance. However, the time and space efficiency of the intelligent RR was not compared to the traditional RR. In this paper, the intelligent RR and traditional RR were compared in terms of time and space complexity using mathematical analysis for nonrecursive algorithms and the asymptotic notation Big-Theta (Big-θ). This is to see if the improvement in system performance affected the efficiency of the algorithm itself. The results show that there was no significant difference in terms of time complexity and no difference in terms of space complexity.

CCS Concepts

• Software and its engineering → Scheduling

Keywords

Scheduling, RR, efficiency

1. INTRODUCTION

1.1 Efficiency in Task Scheduling

Algorithm efficiency is an important concept in the field of computer science, especially when talking about the complexity of programs. Algorithm efficiency affects overall computer efficiency [10].The comparison of different algorithms in terms of either memory space efficiency and runtime efficiency helps in designing economical and cost-efficient algorithms [9], although there have been misconceptions on the topic that may lead to ineffective programming [3] [7].

Another concept that relates to efficiency is resource allocation, specifically task scheduling. Scheduling is the process of determining which processes or tasks to perform first and is crucial in many computer systems [1] [5] [11]. While a scheduling algorithm does not affect a system's behavior, it does have an effect on the system's efficiency as a whole [8]. An ideal scheduling algorithm maximizes CPU utilization which can be seen through different performance indicators such as response time, throughput, and waiting time [2] [4] [6] [11].

There are many types of scheduling algorithms, all with their own advantages and limitations [1] [5] [8]. By combining and hybridizing different scheduling algorithms together, it is possible to create a new and more efficient scheduling algorithm [1]. An example of this is in [6] wherein a new scheduling algorithm for real-time systems was made by combining Shortest Job First scheduling with Earliest Deadline First scheduling to increase success rates and quicken response times.

1.2 Round Robin Scheduling Algorithm

For scheduling, the most commonly and practically used algorithm is the Round Robin (RR) algorithm [11]. The RR has a timesharing characteristic, which is done by distributing an equal amount of processing time to all processes in the queue [8] [9]. The amount of processing time is referred to as the time quantum and determines much of the RR's performance and efficiency [8]. The RR's popularity as a scheduling algorithm has led to the creation of modified RR algorithms for scheduling, mostly by combining it with features of other scheduling algorithms [4] [8] [9] [11].

The intelligent RR algorithm introduced in [11] modifies RR by adding features of the Shortest Remaining Time First algorithm. The research shows that the intelligent RR algorithm had better system performance than the traditional RR algorithm in terms of waiting time, turnaround time and context switches. However, it was not compared to the traditional RR algorithm in terms of time and space efficiency.

This paper aims to compare the intelligent RR algorithm in [11] and the traditional RR algorithm in terms of time efficiency and space efficiency. A comparison of the two algorithms would help in finding out whether the improvements in terms of scheduling criteria (waiting time, turnaround time and context switches) had an effect on the time and space complexity of the algorithm.

2. METHODOLOGY

To compare the time efficiency of the intelligent RR algorithm and the traditional RR algorithm, mathematical analysis for nonrecursive algorithms was first used to compute for the frequency count and order of growth of both algorithms. The asymptotic notation Big-Theta (Big-θ) was then used to compare the two algorithms.

To compare the space efficiency of the intelligent RR algorithm and the traditional RR algorithm, mathematical analysis for nonrecursive algorithms was also used to compute for the memory count and order of growth of both algorithms. For comparison, the asymptotic notation Big-Theta (Big-θ) was also used.

2.1 Calculating the Order of Growth

2.1.1 Mathematical Analysis (Time Complexity)

2.1.1.1 Traditional Round Robin

1. Input size is “n” (number of ELEMENTS in the ARRAY)
   a) Where ELEMENTS are the tasks/processes
   b) Where ARRAY is the ready queue
2. Basic operation is “execute the process for the TQ” or “p[i].execute()”
3. No worst, average, or best cases
4. c(n) $=\sum_{i=0}^{n} 1$ = n - 0 + 1 = n + 1
5. n + 1 $\epsilon\ \theta(n)$
   Order of Growth is “n”

2.1.1.2 intelligent Round Robin

1. Input size is “n” (number of ELEMENTS in the ARRAY)
   a) Where ELEMENTS are the tasks/processes
   b) Where ARRAY is the ready queue
2. Basic operation is “execute the process for the TQ” or “p[i].execute()”
3. No worst, average, or best cases
4. c(n) $=\sum_{i=0}^{n} 1$ = n - 0 + 1 = n + 1
5. n + 1 $\epsilon\ \theta(n)$
   Order of Growth is “n”

As there are no worst, average, or best cases for both algorithms, each can be said to only have the worst cases.

2.1.2 Frequency Count and Memory Count

2.1.2.1 Time Complexity

The following shows the code for the traditional RR algorithm, and beside it are the steps in getting the frequency count. The order of growth was also extracted from the computed frequency count.

```
for (int i = 0; i < n; i ++){                              1 + (n + 1) + n

   if (p[i].BT <= TQ)                                 1 * (1 + (n + 1) + n)

      p[i].execute(BT);            if true

   else                            if false

      p[i].execute(TQ);            if false

}

TOTAL                              if true                          3n + 3

                                   if false                         4n + 4

                                   average                      3.5n + 3.5

                                                        3.5n + 3.5 ϵ θ(n)

                                                      Order of Growth is "n"
```

The following shows the code for the intelligent RR algorithm, and beside it are the steps in getting the frequency count. The order of growth was also extracted from the computed frequency count.

```
 for (int i = 0; i < n; i ++) {                            1 + (n + 1) + n


    if (p[i].BT <= TQ)                                1 * (1 + (n + 1) + n)

       p[i].execute(BT);           if true            1 * (1 + (n + 1) + n)

    else if (p[i].BT <= (TQ        if false           1 * (1 + (n + 1) + n)
 + TQ/2))

       p[i].execute(BT);           if true            1 * (1 + (n + 1) + n)

    else                           if false           1 * (1 + (n + 1) + n)

       p[i].execute(TQ);           if false           1 * (1 + (n + 1) + n)

 }

 TOTAL                             if true                          3n + 3
```

```
                  if false                  4n + 4
                        true

                  if false                  5n + 5
                       false

                  average                   4n + 4
                                       4n + 4 ϵ θ(n)
                                  Order of Growth is "n"
```

2.1.2.2 Space Complexity

The following shows the code for the traditional RR algorithm, and beside it are the steps in getting the memory count. The order of growth was also extracted from the computed memory count.

```
for (int i = 0; i < n; i++) {                       2 + 2

    if (p[i].BT <= TQ)                              2 + 2
        p[i].execute(BT);

    else
        p[i].execute(TQ);
}
TOTAL                                               8
                                            8 ϵ θ(c)

                                  Order of Growth is "c"
```

The following shows the code for the intelligent RR algorithm, and beside it are the steps in getting the memory count. The order of growth was also extracted from the computed memory count.

```
for (int i = 0; i < n; i++) {                       2 + 2
    if (p[i].BT <= TQ)                              2 + 2
        p[i].execute(BT);
    else if (p[i].BT <= (TQ + TQ/2))
        p[i].execute(BT);

    else

        p[i].execute(TQ);

}
TOTAL                                               8
                                            8 ϵ θ(c)
                                  Order of Growth is "c"
```

3. FINDINGS

The following table shows a summary of the computed time and space complexity of the traditional RR algorithm and the intelligent RR algorithm.

Table 1. Summary of Time and Space Complexity

Algorithm	**Complexity**	
	Time	**Space**
Traditional RR Algorithm	3.5n + 3.5 O(n)	8 O(c)
Intelligent RR Algorithm	4n + 4 O(n)	8 O(c)

As seen in the table, there was no significant difference between the time complexities of the two algorithms, and there was no difference at all between the space complexities of the two algorithms. The order of growth for the time and space complexities remained the same in the improved intelligent RR algorithm.

4. DISCUSSION

In terms of time complexity, the intelligent RR algorithm was found to have a higher frequency count than the traditional RR algorithm. The increase in time complexity was due to some additional statements that allowed it to perform better than the traditional RR algorithm in a real-world setting [11]. However, the higher time complexity of the intelligent RR algorithm as compared to the traditional RR algorithm is negligible because both algorithms have the same order of growth - n. Thus, we can conclude that it is possible to significantly improve the performance of an algorithm without increasing its order of growth.

5. CONCLUSIONS

The comparison of the traditional RR and intelligent RR shows no significant difference in time and space efficiency. These results show that the improvement in system performance in the intelligent RR algorithm do not have any detriment on the time and space efficiency of the algorithm itself.

6. ACKNOWLEDGEMENTS

The researchers would like to thank the staff of the College of Information Technology and Computer Science of the University of the Cordilleras for their support.

7. REFERENCES

[1] Arunarani, A. R., Manjula, D., Sugumaran, V. (2018). Task scheduling techniques in cloud computing: A literature survey**.** *Future Generation Computer Systems.* 91 (Feb. 2019), 407 - 415. DOI=10.1016/j.future.2018.09.014

[2] Chan, W., Lam, T., Liu, K., Wong, P. W. H. (2006). New resource augmentation analysis of the total stretch of SRPT and SJF in multiprocessor scheduling.*Theoretical Computer Science.* 359 (2006), 430 - 439. DOI=10.1016/j.tcs.2006.06.003

[3] Gal-Ezer, J., Zur, E. (2004). The efficiency of algorithms—misconceptions. *Computers & Education.* 42 (2004). DOI=10.1016/j.compedu.2003.07.004

[4] Hyytiä, E., Aalto, S. (2016). On Round-Robin routing with FCFS and LCFS scheduling. *Performance Evaluation.* 97 (March 2016), 83 - 103. DOI=10.1016/j.peva.2016.01.002

[5] Kumar, M., Sharma, S. C., Goel, A., Singh, S.P. (2019). A comprehensive survey for scheduling techniques in cloud computing. *Journal of Network and Computer Applications 143 (2019).* DOI=10.1016/j.jnca.2019.06.006

[6] Li, W., Kavi, K., Akl, R. (2007). A non-preemptive scheduling algorithm for soft real-time systems. *Computers and Electrical Engineering. 33 (2007).*DOI=10.1016/j.compeleceng.2006.04.002

[7] Özdener, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer-programming students. *Computers & Education (2008).* DOI=10.1016/j.compedu.2007.10.008

[8] Pradhan, P., Behera, P. K., Ray, B.N.B. (2016). Modified Round Robin Algorithm for Resource Allocation in Cloud Computing. *International Conference on Computational Modeling and Security* (2016). DOI=10.1016/j.procs.2016.05.278

[9] Ramos, J. R., Vernon, R., Sang, J. (2006). An efficient burst-arrival and batch-departure algorithm for round-robin service. *Simulation Modelling Practice and Theory.* 14 (2006). DOI=10.1016/j.simpat.2005.02.008

[10] Ryabko, B. (2012). On the efficiency and capacity of computers. *Applied Mathematics Letters* (2012). DOI=10.1016/j.aml.2011.09.021

[11] Sabha, S. U. (2018). A Novel And Efficient Round Robin Algorithm With Intelligent Time Slice And Shortest Remaining Time First. *ICMMM* (2017). DOI=10.1016/j.matpr.2018.02.17