**Study of the effects on the efficiency of the alterization of Shortest Job First by implementing the Lottery Algorithm**

**Baltazar, John Marli Q.**

**University of the Cordilleras  
Gov. Pack Road, Baguio City,**

**Philippines, 2600  
+63 905 338 7550**

**bjohnmarli@gmail.com**

**Patacsil, Marc Christian C.**

**University of the Cordilleras  
Gov. Pack Road, Baguio City,**

**Philippines, 2600  
661-0125, +63 929 496 2530**

**marc.c.cpatacsil**[**@gmail.com**](mailto:jjpcuyan@gmail.com)

**Osmena, Raymond Rafael S.**

**University of the Cordilleras  
Gov. Pack Road, Baguio City,**

**Philippines, 2600  
424-4501, +63 927 776 8169**

**retzyukino@gmail.com**

**CCS Concepts**

**Software and its engineering → Software organization and properties → Contextual software domains → Operating systems → Process management → Scheduling**

**Abstract**

CPU scheduling is an important factor we all need to understand in properly utilizing the time and efficiency of a job. It has a big effect on the overall performance and resource utilization of the system, since it is of course one of the primary resources of the computer. This research paper will discuss about different kinds of scheduling algorithms such as the Shortest Job First and Lottery Scheduling algorithm, while also focusing on the effects of implementing the Lottery Scheduling Algorithm to the Shortest Job First and identify if the efficiency is better than the original Shortest Job First Scheduling Algorithm. Our research would show the computed efficiency of the Shortest Job First algorithm compared to our algorithm, named as LAWSJF(Lottery Alternating with Shortest Job First). After computing for the computed time complexity and space complexity of both algorithms, we were able to find that the LAWSJF fell short on both results leading us to research upon alternatives. For the final statement, based on the findings we can conclude that as a scheduling algorithm, the LAWSJF algorithm is not recommended when it is based on time and space complexity and that it is much preferred to use the normal Shortest Job First.

**Keywords**

**Scheduling Algorithms; Lottery Scheduling; Shortest Job First; Mathematical Analysis;**

**1. INTRODUCTION**

In most algorithms in their computational complexity it mainly focuses on the time and space complexity of a specific algorithm [1]. In research about scheduling algorithms, a given objective of doing so is to find about the importance of each algorithm to be able to compare the said algorithm with others. Doing so may be able to help in giving out recommendations and improvements this is but to be able to do that, first, we must understand the priority of each scheduling algorithm such as to become less complex, simpler or easier to understand and memory efficient.[2] Algorithms can also be shown or described through many methods such as flowcharts, diagrams, pseudocode, etc. [3] but we will be mainly focusing on showing the algorithm through pseudocode.

CPU scheduling is the basis of multiprogrammed operating systems.[4] Thus, the operating-system design is central to the scheduling algorithm.[5] For different tasks to be executed on the system with a single or multiple processing units, time constraints have been studied and computed. [6] It is also said that the complexity of operating systems that utilize different scheduling algorithms is increasing, a scheduler code is spread on numerous files meaning that the memory is heavily used.[7]

Lottery Scheduling is a randomized resource allocation mechanism. Lottery tickets represent the resource rights of the client and a lottery will happen to determine each allocation [8]. We were able to discover that using randomized priority algorithms example is the lottery scheduling algorithm that can counter the recurring problem of greedy algorithms such as the shortest job first[9].

The shortest job first is a scheduling rule that checks which process with the least amount of execution time [10]. The process will be executed, it is also known as the greedy algorithm, the greedy algorithm is a simple and straightforward algorithm though research says that it would still be better to look for more precise research about said algorithm[11].

# This research paper aims to create a thorough analysis on the Shortest Job First algorithm compared to the proposed algorithm, the LAWSJF. The LAWSJF will wherein utilize both algorithms by alternating shortest job first and lottery algorithm, this research paper will show if the time complexity and space complexity of the LAWSJF will be more time-efficient and memory-efficient than the normal shortest job first algorithm.

# **2. REVIEW OF RELATED LITERATURE**

There are many scheduling algorithms that can be compared to each other by different factors and characteristics, but we will mainly focus on the efficiency of these following algorithms the Shortest Job First & Lottery Scheduling.

As of now, most lottery algorithms are primarily used as a means of resolution to starvation. We plan to implement Lottery Scheduling Algorithm to the Original Shortest Job First and identify the efficiency of the Original Shortest Job First Algorithm compared to our proposed algorithm the LAWSJF. Carl Waldspurger published a study titled ” Lottery Scheduling: Flexible Proportional-Share Resource Management” which talks about the advantages of using the lottery scheduling algorithm and were able to conclude that the lottery scheduling algorithm provides efficient and responsive control over the relative execution rates of computation while also being able to facilitate modular resource management. Since lottery scheduling is conceptually simple and easily implemented, it can be added to existing operating systems to provide greatly improved control over resource consumption rates. Another study is the “Implementing Lottery Scheduling: Matching the Specializations in Traditional Schedulers” by David Petrou which also concludes on the same idea as to the improvement of an algorithm with the implementation of a lottery scheduling [12].

“Equitable Shortest Job First: A Preemptive Scheduling Algorithm for Soft Real-Time Systems” a study done by Rene Mario, which talks about the Shortest Job First algorithm its advantages and that the said algorithm could be incorporated into more complicated scheduling algorithms for ensuring the quality of service in soft real-time systems[13].

# **3. METHODOLOGY**

By doing the mathematical analysis of the LAWSJF, we would first analyze and compute the Shortest Job First algorithm and the Lottery Algorithm since the LAWSJF utilizes both algorithms. This will also compute for the time complexity where the time efficiency will be calculated by how many times the algorithm would run. As for the space complexity, the space efficiency will be calculated by the algorithms data structure. The result of the computation is used to determine the order of growth of the algorithms. The order of growth of the algorithms will then be compared using asymptotic notation. The Big O notation will be used since this notation is frequently used and is more effective when analyzing algorithms. Thus, the researchers will utilize the Big O notation when analyzing and comparing both algorithms in their time complexity and space complexity.

The steps below would show the algorithm of the proposed algorithm or the LAWSJF:

BT = Burst Time.

(LAWSJF)

1. Give each process a number of tickets. (Process # of Tickets = BT/2).
2. Sort each process based on their burst time.
3. Run Shortest Job First.

Case 1:

Shortest job first algorithm

Break;

switch++;

1. Execute process with the least burst time.(The SJF Algorithm). Process[i] is the Burst time of the Process.

for (int i=0; i<n; i++)

{

if (process[i]<min)

{

min=process[i];

c=i;

}

}

Execute process[i] since it has the least BT.

1. Alternate with the Lottery Algorithm.

Case 2:

Lottery Algorithm

Break;

Switch--;

1. Execute process that was chosen as the winner. (Process # of Tickets == 0). Lottery = BT/2.

int winner = rand.nextInt(max\_tick);

for(int i =0;i<n;i++)

for(int z=0;z<lottery[i];z++)

if(ticket[i][z]==winner)

q=i;

Execute process[q] since it has the ticket that was chosen as the winner.

1. Return to step 3.

In a non-recursive algorithm there are given steps in properly analyzing the algorithms math analysis, the first step being, identifying the parameter/s indicating it input size. Second step is determining the basic operation of the algorithm which is if(process[i]<min) for the Shortest Job First algorithm and if(ticket[i][z]==winner) for the Lottery algorithm. The third step is figuring out if the algorithm either has a best, worst and/or average case. For our algorithm there is only one case identified that being the worst case since it is the only case. For step four we will find the calculation of the number of times the basic operation is executed, for there we will derive the order of growth for step five the calculation of the order of growth and frequency count it will be discussed in results and discussion.

# **4. RESULTS AND DISCUSSION**

The computed time complexity for the shortest job first is 3n² +3n+1, where n is the number of times the basic operation is executed. The code compares each process where the process with the least burst time would be executed first. The order of growth for this algorithm is n². If the lottery algorithm would be implemented on the normal shortest jump first, the computed time complexity for the lottery algorithm is 2n³+4n²+5n+3, where n is the number of times the basic operation is executed. The code chooses a random number/ticket from the processes, then the process with that number/ticket will be chosen as the winner and will be executed. The order of growth for this algorithm is n³. In our algorithm, we utilize both algorithms meaning that the time complexity for the code uses both of the computed time complexity of the algorithms. Since the proposed algorithm alternates both the shortest job first and lottery algorithm, their time complexity will be divided by 2. When compared using Big O, the shortest job first algorithm is less than the LAWSJF since the proposed algorithm utilizes both algorithms making it more complex.

The computed space complexity for the standard Shortest Job First is 10+(4n), where n represents the number of processes, making the total space complexity of the standard Shortest Job First is 10 bytes added by the product of 4 bytes multiplied by the number of processes. Where the computed space complexity of the LAWSJF algorithm is 16+ (4n) where n still represents the number of processes. Henceforth the space complexity for the LAWSJF is 16 added by the product of 4 byte multiplied by n. There has been a huge significance to the calculated space complexity, which means that the order of growth is different to the original Shortest Job First.

# **5. CONCLUSION**

Based on the mathematical computations done, the algorithm LAWSJF has become more complex. In the time complexity Since the LAWSJF utilizes both algorithms the Shortest Job First and Lottery Algorithm the computed time complexity resulted to a minor difference as to the original algorithm being more efficient than the modified version that being the LAWSJF, though the results of both are different the order of growth of both algorithms still remains the same.

As for the space complexity we can see that there is a slight difference found as to the increment of the space complexity of the LAWSJF. That being said, it is shown that even though there was a slight increment the order of growth of both algorithms still remains the same as well, for the increment does not affect the number of processes being run. As explained above we can conclude that the altered version was proved to be less efficient compared to the original version.

**6. RECOMMENDATIONS**

As a scheduling algorithm, the LAWSJF algorithm is not recommended when it is based on time and space complexity and that it is much preferred to use the normal SJF. In addition, this research recommends that future researchers try to compare it to other algorithms since this research mainly focus on studying the effects on the shortest jump first algorithm if it implements with the lottery algorithm. It is also possible to try a different approach where the lottery algorithm uses time slices instead to execute the processes, this could change the efficiency of the code.

# **7. REFERENCES**

[1]. Curry, D. M., & Dagli, C. H. (2014). Computational Complexity Measures for Many-objective Optimization Problems. Procedia Computer Science, 36, 185–191. doi:10.1016/j.procs.2014.09.077   
[2]. Letonsaari, M., & Selin, J. (2017). Modeling computational algorithms using nonlinear storytelling methods of computer game design. Procedia Computer Science, 119, 131–138. DOI:10.1016/j.procs.2017.11.169

[3]Papakonstantinou, P. A. (2006). Hierarchies for classes of priority algorithms for Job Scheduling. Theoretical Computer Science, 352(1-3), 181–189. doi:10.1016/j.tcs.2005.10.045

[4]Shoaib M. & Farooqui M.Z.(2014) A Comparative Review of CPU Scheduling Algorithms

[5] Al-Husiainy M. (2007). Best-Job-First CPU Scheduling Algorithm [6]. Zhu, D., Qi, X., Mossé, D., & Melhem, R. (2011). An optimal boundary fair scheduling algorithm for multiprocessor real-time systems. Journal of Parallel and Distributed Computing, 71(10), 1411–1425. doi:10.1016/j.jpdc.2011.06.003

[6]. Zhu, D., Qi, X., Mossé, D., & Melhem, R. (2011). An optimal boundary fair scheduling algorithm for multiprocessor real-time systems. Journal of Parallel and Distributed Computing, 71(10), 1411–1425. DOI:10.1016/j.jpdc.2011.06.003

[7]Maggio, M., Terraneo, F., Papadopoulos, A. V., & Leva, A. (2012). A PI-based control structure as an operating system scheduler. IFAC Proceedings Volumes, 45(3), 329–334. DOI:10.3182/20120328-3-it-3014.00056

[8] Waldspurger. C & Weihl. W (2001) Lottery Scheduling: Flexible Proportional-Share Resource Management (pp. 2) OSDI '94 Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation.

[9]Angelopoulos, S., & Borodin, A. (2010). Randomized priority algorithms. Theoretical Computer Science, 411(26-28), 2542–2558. DOI:10.1016/j.tcs.2010.03.014   
[10] Li, W., Kavi, K., & Akl, R. (2007). A non-preemptive scheduling algorithm for soft real-time systems. Computers & Electrical Engineering, 33(1), 12–29. doi:10.1016/j.compeleceng.2006.04.002   
[11] Shi, C., Zhang, H., & Qin, C. (2015). A faster algorithm for the resource allocation problem with convex cost functions. Journal of Discrete Algorithms, 34, 137–146. DOI:10.1016/j.jda.2015.06.001

[12]David P., John W. M., Garth A. G. (1999) Implementing Lottery Scheduling: Matching the Specializations in Traditional Schedulers

[13]Rene, Mario J. and Kagaris, Dimitri. (2014) "Equitable Shortest Job First: A Preemptive Scheduling Algorithm for Soft Real-Time Systems." INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH AND INNOVATION 6, No. 1