Improving Short Job First adopting a Lottery Scheduling Algorithm focusing on minimizing the average waiting time and average turnaround time

John Marli B. Baltazar

CS Undergraduate

University of the Cordilleras  
Gov. Pack Road, Baguio City   
Philippines, 2600   
Telephone number, +63

[bjohnmarli@gmail.com](mailto:bjohnmarli@gmail.com)

Raymond Rafael S. Osmeña

CS Undergraduate

University of the Cordilleras  
Gov. Pack Road, Baguio City   
Philippines, 2600   
Telephone number, +63

[retzyukino@gmail.com](mailto:retzyukino@gmail.com)

Rey Benjamin M. Baquirin

MSCS

University of the Cordilleras  
Gov. Pack Road, Baguio City   
Philippines, 2600   
Telephone number, +63

[reybenbaq@gmail.com](mailto:reybenbaq@gmail.com)

Marc Christian C. Patacsil

CS Undergraduate

University of the Cordilleras  
Gov. Pack Road, Baguio City   
Philippines, 2600   
Telephone number, +63

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

**ABSTRACT**

Shortest Job First is a scheduling algorithm in which the process with the smallest execution time will be executed first, the disadvantage of this scheduling algorithm is that longer processes will have more waiting time in which they’ll suffer starvation. Our idea of enhancing short job first with the use of lottery scheduling algorithm this can solve the disadvantages of short job first which is the starvation of processes with the use of lottery which can solve it by giving processes with higher burst time a chance to get chosen to be run and the enhance by minimizing the AWT and TAT of the processes. Results shown on this paper that the Lottery Alternating with Short Job First (LAWSJF) improved the original Short Job First by reducing average waiting time and average turn around time in the test cases.

**CCS Concepts**

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

**Keywords**

Scheduling, CPU, Algorithm, Operating System, CPU Scheduling, Shortest Job First, Lottery Scheduling Algorithm.

# INTRODUCTION

CPU scheduling is an important factor we all need to understand in properly utilizing the time and efficiency of a job. CPU scheduling is the basis of multi programmed operating systems[1] and it has a big effect on the overall performance and resource utilization of the system [2].The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design.[3]. The aim of CPU scheduling is to make the system efficient, fast and fair. Scheduling is a fundamental operating-system function[4], it also refers to a set of rules, policies and mechanisms that govern the order in which resources are allocated to the various processes and the work is to be done[5].The CPU is, of course, one of the primary computer resources[6]. Thus, its scheduling is central to operating-system design. CPU may contain different cores or single core and one core handle one process at a time[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]. The criteria of Shortest Job First (SJF) scheduling algorithm is that it process the smallest task first[9].Along with minimum average waiting time, Shortest Job first introduces starvation for bigger processes, which is a strong shortcoming of this smart technique[10], hence the implementation of the Lottery Scheduling algorithm does not need to be uniform as to its purpose is to solve starvation.[11]

This paper focuses on the improvement of the Shortest Job First algorithm where-in we will be enhancing the SJF algorithm by using Lottery scheduling algorithm. By implementing both algorithms and by alternating Shortest job first and lottery, this research paper will show the Shortest Job First algorithm would be improved on the given metrics.

# RELATED LITERATURE

There are many scheduling algorithms that can be compared to each other by different factors and characteristics. To properly be able to compare an algorithms performance the Users must understand the many criteria of a CPU Scheduling algorithm. The following below are possible criteria:

**CPU utilization**: To make out the best use of CPU and to not waste a CPU cycle we want to keep the CPU as busy as possible. It would be ideal for the CPU to be working at 100% of the time. But Considering a real system, CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)

**Throughput**: It is the total number of processes completed per unit time or rather say total amount of work done in a unit of time. This may range from ten second to one hour depending on the specific processes.

**Turnaround Time**: From the submission time of a process to its completion time is turnaround time. It refers to the total time period spends waiting to get into memory, waiting in the ready queue, executing in the CPU and doing input/output operations.

**Waiting Time**: The sum of the periods spent waiting in the ready queue amount of time a process has been waiting in the ready queue to acquire get control on the CPU.

**Response Time**: It is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

**Context Switch**: The process of storing the state of a process or of a thread, so that it can be restored and execution resumed from the same point later. This allows multiple processes to share a single CPU, and is an essential feature of a multitasking operating system.

**Load Average**: It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.

As of now most lottery algorithms are primarily used as a means of resolution to starvation. But for us we plan to utilize the lottery algorithm which mainly focuses on the CPU usage and alternate it with a Shortest Job First algorithm so that we can enhance the overall algorithms waiting time but by also solving the starvation in which is found in shortest Job first algorithms.” Lottery Scheduling: Flexible Proportional-Share Resource Management” by Waldspurger Carl and “Implementing Lottery Scheduling: Matching the Specializations in Traditional Schedulers” by David Petrou, are some of these studies also talk about how efficient of the Lottery Scheduling algorithm. While “Equitable Shortest Job First: A Preemptive Scheduling Algorithm for Soft Real-Time Systems” by Rene Mario, is a study which talks about the situation on Shortest Job First and its starvation problem.

## METHODOLOGY

**(1)** Each process will have burst time and arrival time as a factor.

**(2)** Processes will then be arranged based on their burst time.

**(3)** Assign tickets to each process, the process with the large burst time would be given a large number of tickets.

**(4)** Once each processes have assigned number of tickets, randomly generate one winning ticket.

**(5)** Execute the process with the shortest burst time.

**(6)** Alternating with Step 5, Execute process where the process number of tickets is 0.

**(7)** Repeat the 5th and 6th steps until all process have been completed.

## PSEUDO CODE

**1**. Enter the number of processes and assign the burst time of each processes.

**2.** Sort each processes in ascending order based on their burst time.

**3.** If there are processes with the same burst time, the one with the higher arrival time will go first.

**4**. Assign tickets to each process where a process max ticket is (max tickets = burst time/2).

**5**. Generate a random ticket.

**6.** Execute process with shortest burst time.

**7**. After execution execute the process that was chosen as the winner based on the generated random number.

**8**. Alternate step 6 and 7 until all process have been executed.

**9**. Repeat step 5-7 until all process are executed.

## 4. Findings

**Illustration of the LAWSJF Algorithm**

Table 1. Sample Process

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst**  **Time** |
| P0 | 1 | 10 |
| P1 | 8 | 14 |
| P2 | 10 | 5 |
| P3 | 14 | 18 |
| P4 | 16 | 20 |
| P5 | 20 | 8 |

Each process would be given a number of tickets based on their burst time, where Process burst time is divided by 2 (Burst Time/2) which the result is the number of tickets of the process, the given quantum in the sample process is 10 which if a process ticket was chosen then subtract the quantum to the process burst time(Quantum - Burst time). SJF would run first followed by lottery which then Process P2 would run first since P2 has the shortest burst time. Since there is no other process that has the same burst time as P2, then P2 would run first. After P2 was executed the lottery would choose a winner among the processes which is random, in this case Process P0 would be executed first since Process P0 was chosen as the winner. After Process P0 was executed it would be followed by the lowest Process which is Process P5 since Process P2 was already executed. The alternating of SJF and Lottery would continue until there is no remaining process to execute.

**Test Cases :**

**Test Case 1:**  We assumed 10 processes.Processes with low burst time and high arrival time (Burst Time < Arrival Time)using these processes to help clarify whether our algorithm will yield better results than the average shortest job first (SJF).

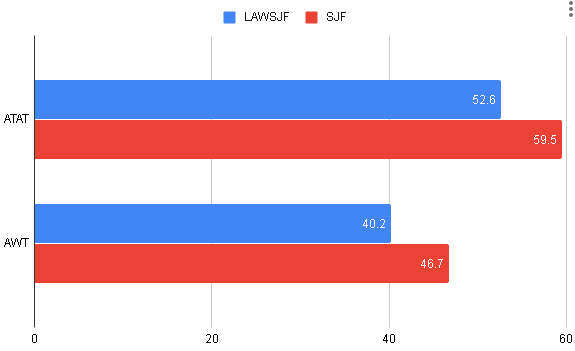
Table 2. Test Case 1

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst**  **Time** |
| A | 20 | 10 |
| B | 22 | 11 |
| C | 25 | 12 |
| D | 26 | 13 |
| E | 28 | 14 |
| F | 21 | 15 |
| G | 24 | 16 |
| H | 23 | 17 |
| I | 27 | 18 |
| J | 29 | 19 |

Table 3. Shows the comparative results of LAWSJF compared to SJF

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **ATAT** | **AWT** |
| LAWSJF | 50.8 | 38.6 |
| SJF | 67 | 52.5 |

Figure 1. Comparative results of Test case 1



**Case 1 Results:** Here in this bar graph we can see that the Average Turn-around time (ATAT) of our Lottery alternating with shortest job first (LAWSJF) was lower than the ATAT of the average Shortest Job First (SJF) at the same time the average Waiting Time (AWT) of the LAWSJF was also lower than the average SJF showing us that the LAWSJF was more efficient in both aspects. Given that the Processes have lower burst time than the arrival time (Burst Time < Arrival Time).

**Test Case 2:**  We assumed 10 processes.  Processes with higher burst time and lower arrival time(Burst Time > Arrival Time) using these processes to help clarify whether our algorithm will yield better results than the average shortest job first (SJF).

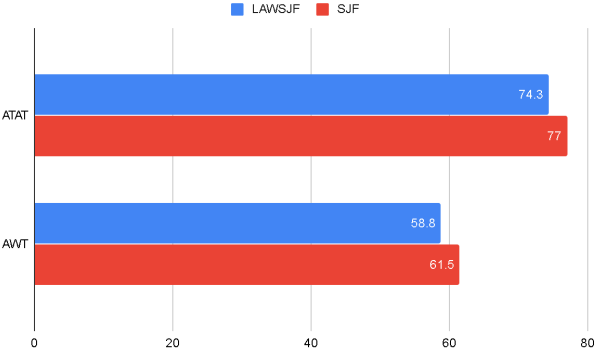
Table 4. Test Case 2

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst**  **Time** |
| A | 1 | 20 |
| B | 4 | 12 |
| C | 5 | 17 |
| D | 7 | 11 |
| E | 2 | 13 |
| F | 10 | 18 |
| G | 6 | 19 |
| H | 3 | 14 |
| I | 8 | 15 |
| J | 9 | 16 |

Table 5. Shows the comparative results of LAWSJF compared to SJF

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **ATAT** | **AWT** |
| LAWSJF | 74.3 | 58.8 |
| SJF | 77 | 61.5 |

Figure 2. Comparative results of Test case 2



**Case 2 Results:** Here in this bar graph we can see that the Average Turn-around time (ATAT) of our Lottery alternating with shortest job first (LAWSJF) was higher than the ATAT of the average Shortest Job First (SJF) at the same time the average Waiting Time (AWT) of the LAWSJF was also higher than the average SJF showing us that the LAWSJF was more efficient in both aspects.  Given that the Processes has large burst times with lower arrival times (Burst Time > Arrival Time).

**Test Case 3:** We assumed 10 processes.  Processes with randomly given burst time and randomly given arrival time using these processes to help clarify whether our algorithm will yield better results than the average shortest job first (SJF).

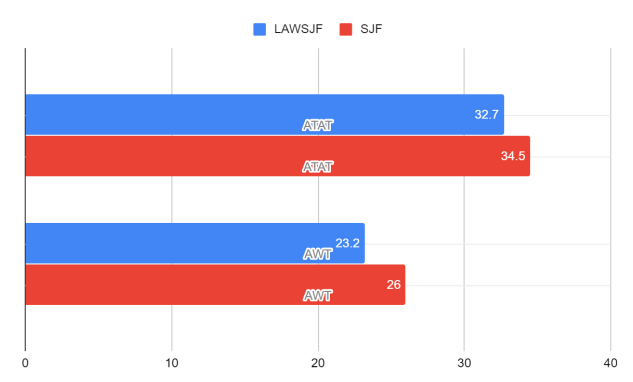
Table 6. Test Case 2

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst**  **Time** |
| A | 1 | 12 |
| B | 15 | 7 |
| C | 3 | 10 |
| D | 6 | 5 |
| E | 2 | 2 |
| F | 10 | 9 |
| G | 5 | 6 |
| H | 8 | 8 |
| I | 11 | 11 |
| J | 9 | 15 |

Table 7. Shows the comparative results of LAWSJF compared to SJF

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **ATAT** | **AWT** |
| LAWSJF | 32.7 | 23.2 |
| SJF | 34.5 | 26 |

Figure 3. Comparative results of Test case 3

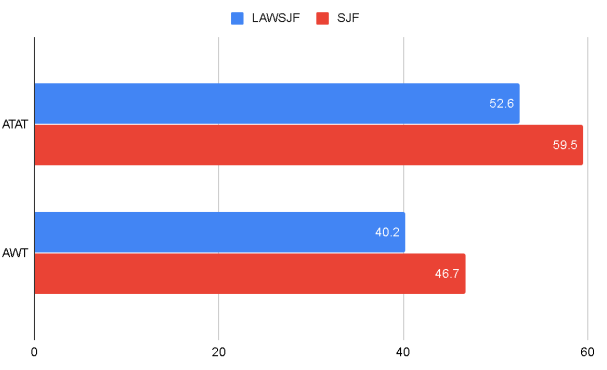


**Case 3 Results:** Here in this bar graph we can see that the Average Turn-around time (ATAT) of our Lottery alternating with shortest job first (LAWSJF) was lower than the ATAT of the average Shortest Job First (SJF) at the same time the average Waiting Time (AWT) of the LAWSJF was also lower than the average SJF showing us that the LAWSJF was more efficient in both aspects. Given that the Processes have random arrival time and burst time.

Table 8. Shows the average comparative result of LAWSJF and SJF.

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **ATAT** | **AWT** |
| LAWSJF | 52.6 | 40.2 |
| SJF | 59.5 | 46.7 |

Figure 4. The average ATAT and AWT of the test cases.



**Summary of the Findings**: As shown in all the figures above we can see and deduct that our algorithm the Lottery alternating with Shortest job first (LAWSJF) came up superior or better than the average Shortest job first yielding better results whether having a random burst time and a given arrival time, whether both factors were random, if the process have  higher arrival time than their burst time or if the process have lower arrival time than their burst time the LAWSJF yielded better results than the average SJF.

# 5. Conclusion and Discussions

**Discussions :**

Based on the Test Cases above, the Lottery alternating with shortest job first scheduling algorithm improves CPU scheduling in two metrics: the average turnaround time and average waiting time. The average turnaround time and average waiting time improvements are shown on all the cases where they both showed a significant amount of improvement compared to an average shortest job first scheduling algorithm. This results were shown whether the burst time was randomized and whether the arrival times are also randomized, the LAWSJF performed better than the original SJF. The results on the test cases where the processes burst time was higher than their arrival time and the processes arrival time was higher than the burst time it is shown that the LAWSJF showed better results than the SJF. The results also shows that the ATAT and AWT of both algorithms are close since the LAWSJF uses the SJF algorithm.

**Conclusion :**

Using the test cases shown above, comparing the results of each to find that the algorithm LAWSJF reduce the average turnaround time and the average waiting time compared to an average SJF. Not only does this algorithm reduce the ATAT and AWT but it was able to solve the typical starvation problem of an SJF by giving chances to other process to be run.

# 5.REFERENCES

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

[2] Goel. N & Garg. R. B (2012) A Comparative Study of CPU Scheduling Algorithms (pp. 245) International Journal of Graphics & Image Processing.

[3] Al-Husiainy M. (2007). Best-Job-First CPU Scheduling Algorithm

[4] Shweta J. & Saurabh J.(2015) A Research Survey and Analysis for CPU Scheduling Algorithms using Probability-Based Study.

[5 ]Shweta J. & Saurabh J.(2016) A Review Study on the CPU Scheduling Algorithms.

[6] Silberschatz, A.    P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th edition, Wiley India

[7] Fayyaz S. , Ali Hamza H. , Moin U Din S. , Riaz E.(2017) Comparative Analysis of Basic CPU Scheduling Algorithms(pp. 43)International Journal of Multidisciplinary Sciences and Engineering.

[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] Alworafi, Mokhtar & Dhari, Atyaf & Al-Hashmi, Asma & Darem, Abdul Basit & Suresha,. (2016). An Improved SJF Scheduling Algorithm in Cloud Computing Environment. 10.1109/ICEECCOT.2016.7955216.

[10] Shahazad B. & Muhammad T.(2005) Optimized Solution to Shortest Job First by eliminating Starvation

[11] Y.A. Adekunle , Z.O. Ogunwobi , A. Sarumi Jerry , B.T. Efuwape , S. Ebiesuwa , and J.P.l Ainam(2014) A Comparative Study of Scheduling Algorithms for Multiprogramming in Real-Time Systems (pp. 183) Innovative Space of Scientific Research Journals.

Columns on Last Page Should Be Made As Close As Possible to Equal Length

Authors’ background

|  |  |  |  |
| --- | --- | --- | --- |
| Your Name | Title\* | Research Field | Personal website |
|  |  |  |  |
|  |  |  |  |

\*This form helps us to understand your paper better, the form itself will not be published.

\*Title can be chosen from: master student, Phd candidate, assistant professor, lecturer, senior lecturer, associate professor, full professor, research, senior research