**The Reduction of Starvation in the Shortest Job First (SJF) Scheduling Algorithm by applying Modified Time Slice**

Rey Benjamin M. Baquirin

MSCS

Governor Pack Road, Baguio City, Philippines 2600

Telephone number, incl. country code

reybenbag@gmail.com  
  
Samuel Joshua Likigan  
CS Undergraduate  
Governor Pack Road, Baguio City, Philippines 2600

Telephone number, incl. country code

jasuel321@gmail.com

Haiwatha Edenreign Saturno  
CS Undergraduate  
Governor Pack Road, Baguio City, Philippines 2600

Telephone number, incl. country code

liliasinth@gmail.com

Rozion Dre Cacho

CS Undergraduate  
Governor Pack Road, Baguio City, Philippines 2600

2nd line of address

Telephone number, incl. country code

RizaArisato@gmail.com

Brandon Casas

CS Undergraduate  
Governor Pack Road, Baguio City, Philippines 2600

2nd line of address

Telephone number, incl. country code

bc5475@gmail.com

**ABSTRACT**

The Shortest Job First (SJF) Algorithm handles a set of processes according to the least amount of execution time, and is thus applicable if the processor knows how much time a process would consume – which isn’t necessarily feasible across all cases. SJF also suffers from an inherent starvation problem for programs lower down the execution line. In contrast to this, Round Robin (RR) Scheduling is far more optimal in terms of time sharing among several processes because it divides an equal amount of time among them. Using a modified version of this technique from RR, which may also be called Time Slice, we try to improve the SJF algorithm in order to resolve its problems with starvation that the algorithm have shown no improvement with both of the cases because the changes observed were negligible differences

**CCS Concepts**:

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

**Keywords**

Scheduling Algorithms; Shortest Job First;

# INTRODUCTION

CPU scheduling, a method used in balancing the amount of computing resource a process can have. As the need for a more efficient and faster scheduling algorithm is needed as the growth of technology requires a better algorithm in order to maximize the use of a CPU. Though as effective as the algorithm can be it still has its problems. Some of which are either CPU utilization, turnaround time, response time, waiting time and throughput.

The researchers aim to improving an existing algorithm. The algorithm to be improved is the Shortest Job First (SJF). This algorithm is defined as a process with the shortest burst time or duration gets to go first.

It can either be a non-preemptive or a preemptive process. It is an effective algorithm that will reduce waiting time. This algorithm is good but since it focuses on the shortest time or the process that have small execution time the process with a longer time will not be focused which will then lead to starvation of CPU process for those with longer time. Wanting to fix the problem, the

researchers will be combining two other methods that have been done by two previous researches.

# REVIEW OF RELATED LITERATURE

CPU Optimization Criteria:

CPU Utilization – making the CPU to at least continue working, the ideal would be working at most times (100%). For a real system, the usage for the CPU may range from 40% (lightly loaded) to 90% (heavily loaded).

Throughput – If the CPU is busy, then it is executing a process. It is the total number of processes completed per unit time or in other words, the total amount of work done in a unit of time. It can range from 10/second to 1/hour depending on the specific processes.

Turnaround Time - amount of time to complete a task from start to finish. Another way to say it is, from the submission of the process to the time of completion of the process (Wall clock time).

Waiting Time - the sum of periods spent waiting in the ready queue, the amount of time a process has been waiting in order to get control of the CPU.

Response Time - the amount of time it takes, starting from the time that a request was sent up until the first response is produced. It is the time that a process starts responding and not the completion of the entire system.

A lot of researches so far has been improving on algorithms be it First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), etc. or it would be combing two other algorithms together. The goal by doing these is to improve on where the algorithm went wrong. Summarizing the problem, First Come First Serve (FCFS) would not focus on burst time, Shortest Job First (SJF) would not let the process with the longest burst time get any CPU and Round Robin mostly relies on time quantum. This would lead to researches in order to help solve the problem either by combing two algorithm concepts with the other or improve/modify it.

The researches have taken interest in the SJF considering that it is an optimal algorithm but the problem of the burst time would not be possible. Though there was researches that would help with solving the problem. The researchers would combine two interesting researches about the SJF. One would be time slice and the other would be comparing the sum with the longest burst time.

Summarizing these concepts starting with the time slice. Time slice would take the shortest burst time and longest burst time and get their average. The other concept would work on two-pointers one for the shortest burst time and the other is for the longest, as you add along in the process the shortest pointer would compare it with the longest if the shortest is greater than or equal with the longest then it would be added next if not then add the next shortest.

# METHODOLODY

The methodology of the algorithm is as follows:

# Adapting the original SJF formula that takes into account Burst Time (BT) to execute process in descending order.

# Arrival Time is assumed to be 0 for testing purposes.

# Processes are inserted into a single queue, if a new process enters the queue, the algorithm will adapt the First Come, First Serve (FCFS) technique.

# Queued processes are arranged according to burst time starting with the lowest.

# The usage of Time Slice on which is adapted from the Round Robin (RR) Algorithm’s Quantum Time mechanic.

# The formula is as follows:

## Determine the sum of process with the shortest and the process with the largest BT.

## Divide the sum by 2 to get TS

# Application of Greater Than concept into Time Slice. The formula is as follows:

## Start with 0. Add process with least burst time

## Determine if the the summation of the next queued process is greater than the TS. If so, add to the process with the greatest burst time to the queue.

## Else, continue adding the next queued process using the same condition until all processes within the queue are accounted for.

# Analysis of the comparison of original SJF to calculated TS

## Pseudocode of Modified SJF Algorithm

Let P(n) be a process to be executed. Let BT be burst time. Let TS be Time Slice.

1. Add P1, P2, P3, and P4 to a queue.
2. Assign BT to P(n).
3. Sort queue in ascending order according to BT.
4. Set TS = Highest BT + Lowest BT / 2
5. Start with 0. Add Lowest BT to 0.
6. If the summation of the process >= to the TS then add the process with the greatest to the queue
7. Add next greatest BT according to sorted queue.

## Test Cases

These tests were done under the following assumptions:

* 1. Processes are executed in a single processor.
  2. The processes are CPU bound.
  3. Burst time is known before execution starts.
  4. SJF, ASJF and ESJF are used as benchmark algorithm.
  5. For the ASJF Arrival time will be used for this algorithm, though for the other three Arrival time will not be used.
  6. The SJF used will be non-preemptive.

### Test Case 1

Table 1. In this test, it will involve SJF, ESJF and TSCSJF. Assuming that arrival time is 0.

|  |  |
| --- | --- |
| **Process** | **Burst Time** |
| P1 | 8 |
| P2 | 6 |
| P3 | 3 |
| P4 | 7 |

### Test Case 2

Table 2. In this test, it will involve SJF, ASJF and TSCSJF. SJF and ASJF will be using arrival time.

|  |  |  |
| --- | --- | --- |
| **Process** | **Burst Time** | **Priority** |
| P1 | 86 | 7 |
| P2 | 54 | 5 |
| P3 | 49 | 4 |
| P4 | 34 | 9 |
| P5 | 92 | 2 |

### Test Case 3

Table 3. In this test, it will involve SJF, ESJF and TSCSJF. Assuming that arrival time is 0.

|  |  |
| --- | --- |
| **Process** | **Burst Time** |
| P1 | 7 |
| P2 | 19 |
| P3 | 6 |
| P4 | 20 |
| P5 | 4 |
| P6 | 2 |

# FINDINGS AND RESULTS

### Test Case 1

Table 4. Through SJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 16 | 24 |
| 3 | 9 |
| 0 | 3 |
| 9 | 16 |

Average Waiting Time: 7

Average Turnaround Time: 13

Table 5. Through ESJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 17 | 24 |
| 3 | 9 |
| 0 | 3 |
| 9 | 17 |

Average Waiting Time: 7.25

Average Turnaround Time: 14

Table 6. Through TSCSJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 9 | 17 |
| 3 | 9 |
| 0 | 3 |
| 17 | 24 |

Average Waiting Time: 7.25

Average Turnaround Time: 14

### Test Case 2

In the concept of ASJF will be getting the average of the shortest and longest burst -time.

TS = (Shortest BT + Longest BT)/2. In here the time slice is 63, (34+92)/2=63.

Table 7. Through SJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 195 | 281 |
| 141 | 195 |
| 92 | 141 |
| 281 | 315 |
| 0 | 92 |

Average Waiting Time: 141.8

Average Turnaround Time: 204.8

Table 8. Through ASJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 166 | 286 |
| 112 | 166 |
| 63 | 112 |
| 229 | 263 |
| 0 | 315 |

Average Waiting Time: 114

Average Turnaround Time: 228.4

Table 9. Through TSCSJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 229 | 315 |
| 175 | 229 |
| 34 | 83 |
| 0 | 34 |
| 83 | 175 |

Average Waiting Time: 104.2

Average Turnaround Time: 167.2

### Test Case 3

Table 10. Through SJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 12 | 19 |
| 19 | 38 |
| 6 | 12 |
| 38 | 58 |
| 2 | 6 |
| 0 | 2 |

Average Waiting Time: 12.83

Average Turnaround Time: 22.5

Table 11. Through ESJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 12 | 19 |
| 19 | 38 |
| 6 | 12 |
| 38 | 58 |
| 2 | 6 |
| 0 | 2 |

Average Waiting Time: 12.83

Average Turnaround Time: 22.5

Table 12. Through TSCSJF

|  |  |
| --- | --- |
| **WAITING TIME** | **TURNAROUND TIME** |
| 32 | 39 |
| 39 | 58 |
| 6 | 12 |
| 12 | 32 |
| 2 | 6 |
| 0 | 2 |

Average Waiting Time: 15.17

Average Turnaround Time: 24.83

# DISCUSSION OF FINDINGS

As the shown in the findings, there are no improvements to waiting time and turnaround time compared to non-preemptive SJF. The three test cases have shown the results of that comparing it with each other have shown that TSCSJF have shown no improvements.

In the case of the average waiting times, the findings showed the TSCSJF algorithm is faster than the SJF in the second case due to the nature of preemptive SJF algorithm.The changes in the order of the processes depending on the least and highest burst times depicts processes after the time slice are then taken into account and putting them ahead of other processes with the implementation of time slice, reducing their waiting times.

In terms of average turnaround times, the findings depict the TSCSJF algorithm is longer compared to the SJF with the exemption of the second case, ESJF and ASJF algorithms. The changes occurred to the burst times of the processes as stated earlier, this would result in longer times for queued processes to be finished due to delaying the turnaround time of the processes with the high burst times. This results in a slight delay in execution of other processes depending on the size of the time slice.

In the first case and third, the average waiting time and turnaround of TSCSJF are slower compared to the SJF algorithm because of the largest process being handled first before the rest of the other processes.

# CONCLUSION

The SJF algorithm is accounted for its simplicity and ease of implementation, but it faces problems relative to starvation and dynamism. With the tests done, the researchers have concluded that the new algorithm has shown some improvement with the cases because the changes observed provided significant differences across the test cases in comparison to the ASJF and ESJF, alongside a solution to the starvation inherent to SJF. The researchers recommend additional study of possible dynamic interactions between the algorithm and incoming processes, as is noted in other scheduling algorithms such as Round Robin with an inherent Quantum Time mechanic, alongside any supplemental method to reduce and waiting time turnaround time and an increase to possible test cases.

# REFERENCES

1. Aggarwal, H., & Nagpal, S. (2014, June). Augmented SJF algorithm with reduced starvation. Retrieved from <http://ijarcsse.com/Before_August_2017/docs/papers/Volume_4/6_June2014/V4I6-0223.pdf>.
2. Kataria, A. (2018, July 28). CPU Scheduling: Operating System. Retrieved from <https://www.includehelp.com/operating-systems/cpu-scheduling.aspx>.
3. Liu, Y. (n.d.). Reading Materials for CPU Scheduling. Retrieved from <https://www.u-aizu.ac.jp/~yliu/teaching/os/lec10r.html>.
4. Studytonight. What is CPU Scheduling? Retrieved from <https://www.studytonight.com/operating-system/cpu-scheduling>.
5. Shahzad, B., & Afzal, M. T. (n.d.). OPTIMIZED SOLUTION TO SHORTEST JOB FIRST BY ELIMINATING THE STARVATION. Retrieved from <https://www.researchgate.net/publication/234556241_OPTIMIZED_SOLUTION_TO_SHORTEST_JOB_FIRST_BY_ELIMINATING_THESTARVATION>.

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

Authors’ background

|  |  |  |  |
| --- | --- | --- | --- |
| Rey Benjamin M. Baquirin | Master Student | Computer Science | N/A |
| Samuel Joshua Likigan | Student | Computer Science | N/A |
| Haiawatha Edenreighn Saturno | Student | Computer Science | N/A |
| Rozion Dre Cacho | Student | Computer Science | N/A |
| Brandon Casas | Student | Computer Scince | N/A |

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