Different Scheduling Algorithms with Comparative Analysis

Chilaka Dhanya Sri, Priya Sahu, Tanay Anand
B20AI007, B20AI031, B20AI046

(PCS-II Course Project)

Abstract- This paper reports our experience with building different scheduling algorithms. This paper will also report the comparative analysis of all the algorithms and their analytic visualization. Where we will compare the various average times of different processes to come to a conclusion.

~Introduction~

A **Scheduling Algorithm** is an algorithm which tells us how much CPU time we can allocate to the processes.

These scheduling algorithms are either preemptive or non-preemptive. **Preemptive Scheduling Algorithms** are those which are based on the priority of the processes. By preference, when a high priority process enters, it preempts a low priority process in between and executes the high priority process first.
Non-preemptive Scheduling Algorithms are those that can't be preempted in between, i.e. we can not take control of the CPU in between until the current process completes its execution.

Why Scheduling Algorithms?

Since a process needs CPU time and I/O time both for its execution. In a multiprogramming system, one process can use the CPU while another process is waiting for I/O whereas, on the other hand in a uni programming system, all the time gets wasted in waiting for I/O while the CPU is free during that time.

The Scheduling performance can be analyzed on the following criteria:

1. **CPU utilization:-**The maximum use of the CPU when it is busy.
2. **Throughput:-** It is the number of processes that complete their execution per unit time.
3. **Turnaround Time:-**It is the amount needed for the execution of a single process.
4. **Waiting Time:-** It is the amount of time a process waits in the ready queue.
5. **Response Time:-**This is the amount of time takes from when a request was submitted until the first response is produced not output.

Dataset:
We have created a dataset with the initialisation of a number of processes and generated arrival time, burst time and priority. Thus, for each of the processes we have three features i.e. arrival time, burst time and priority.

~Methodology~

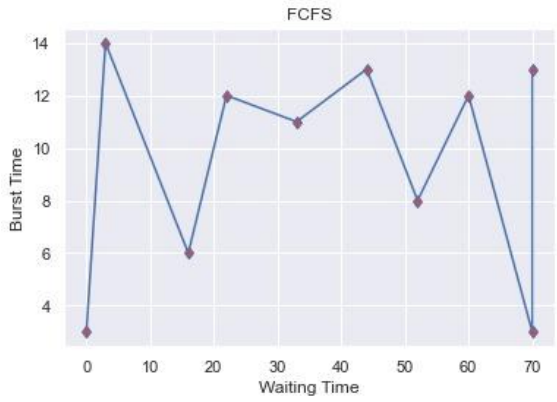
Overview
There are various scheduling algorithms present out of which we shall implement the following

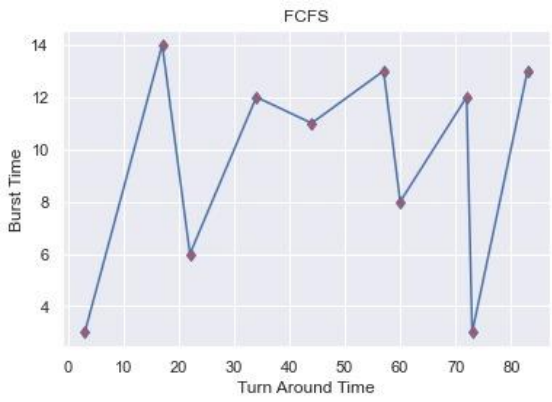
- FCFS - First come first serve
- SJF(non-preemptive) - Shortest Job First
- Priority Scheduling (Pre-emptive)
- SJF(pre-emptive) - Shortest Job First
- Round Robin

Dataset Creation:
For this part, we initialized the number of processes(array) and then generated burst time, arrival time and priority for each of the processes. Then sorted the processes according to the arrival time and assigned Pid to every process.

The scheduling algorithms:
FCFS - First come first serve
FCFS is a non-preemptive scheduling algorithm. It uses the First in-First out) FIFO strategy to assign the priority to processes in the order, which is the same as the request made by the process for the processor. The process or job that requests the CPU first is allocated the CPU first and other in the queue has to wait until the CPU is free. FCFS is also known as First-In-First-Out (FIFO), it is the simplest scheduling methodology. All the later arriving jobs are inserted into the tail (rear) of the ready queue and the process to be executed next is removed from the head (front) of the queue and the control of the current process is transferred to the CPU. FCFS gives better performance for longer jobs and fewer multiple processes in the ready queue. The relative importance of jobs is measured only by arrival time (poor choice).

The drawback of FIFO can be observed as the average time for waiting for a purely FIFO system is very poor.





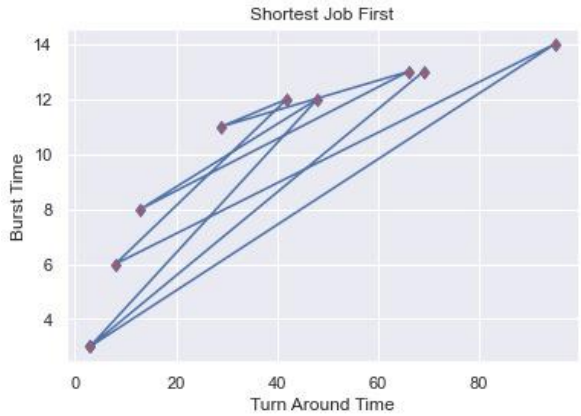
~SJF~

In the SJF technique, the shortest amongst the entire ready queue job is executed first rest are preempted. The benefit of this is that waiting time is minimal for the shorter jobs. The SJF is especially appropriate for the batch jobs for which the run time is known in advance. SJF is the FCFS concept, but the process which comes at the same time or the same ready queue must be sorted in ascending order. The smallest burst time will be the first order while the bigger will be the last one. The process table must be reconstructed in the SJF model. The same arrival time must be sorted from low to high.

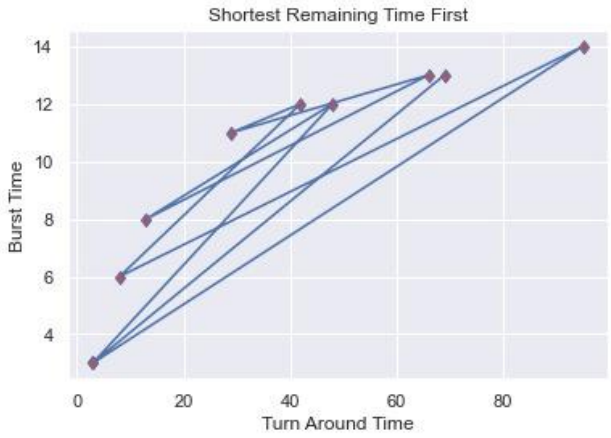
The drawback of the SJF algorithm is knowing which incoming process is indeed shorter than another. This requires a separate algorithm running for monitoring and sorting the jobs in real-time. Also, long-running jobs may starve, because the CPU may have a good and steady supply of short jobs.

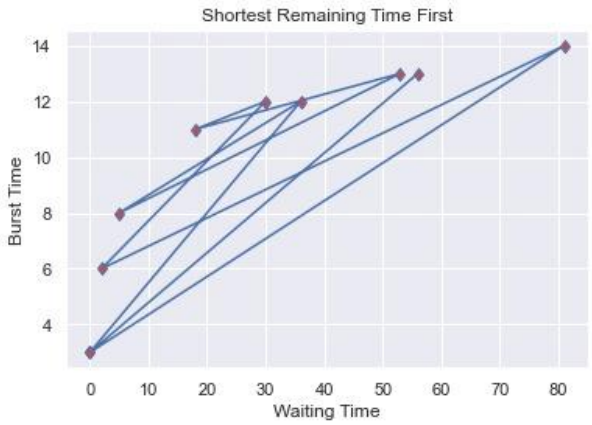
SJF(non-preemptive) - Shortest Job First

Once the CPU is given to the process it cannot be preempted until completes its CPU burst, even though the arriving process has a shorter burst time.



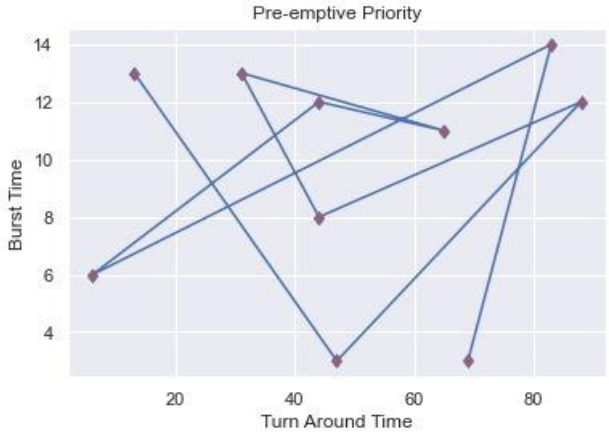
SJF(pre-emptive) - Shortest Job First -(Shortest Remaining Time First)
If a new process arrives at the CPU with a shorter burst time than the currently executing process then the CPU preempts the currently executing time of the currently executing process. This scheme is also known as the shortest remaining time first (SRTF).

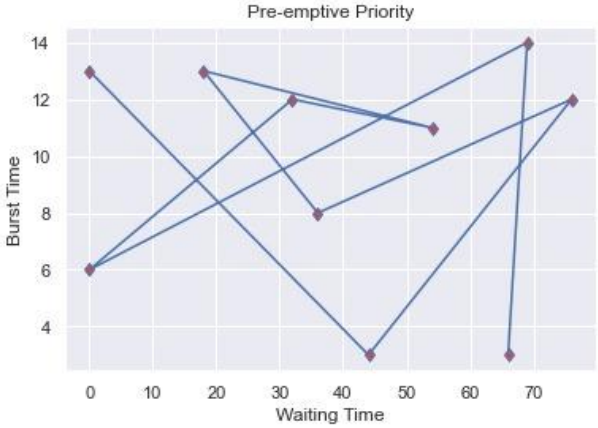




Priority Scheduling (Pre-emptive)
In the Priority scheduling algorithm, each process is assigned priority by either an outer agency or as per their system requirements and as soon as each process hits the queue it is sorted based on its priority so that processes with higher priority are dealt with first. In case two processes arrive with the same priority in a different order then they are executed in FCFS order. The main advantage of Priority scheduling is that the important jobs can be finished first.

The drawback with Priority scheduling is when the operating system gives a particular task very low priority so it sits in the queue for a larger amount of time, not being dealt with by the CPU.

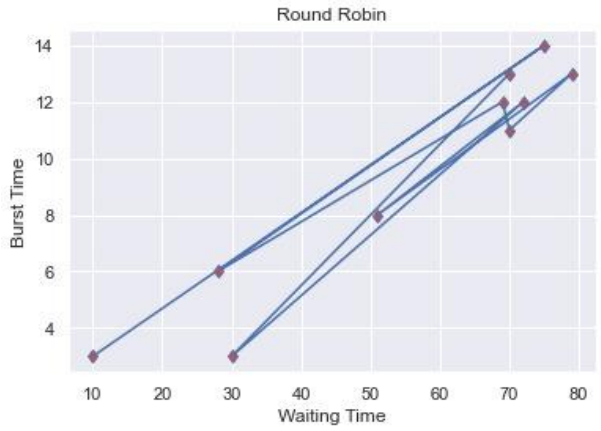


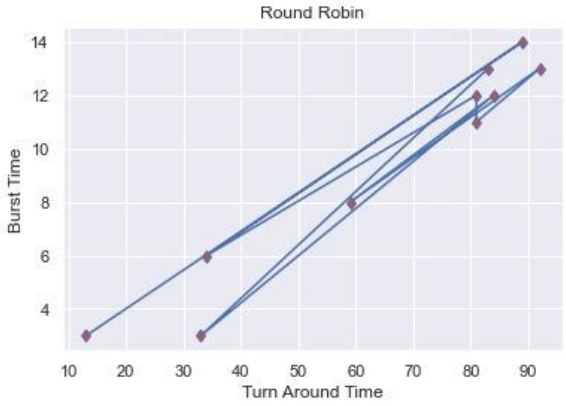


Round Robin

In this approach, a fixed time slot is defined before the execution of processes starts, which is a normal small unit of time. In each time slice (quantum) the CPU executes the current process only up to the end of the time slice. If that process is having less burst time than the time slice then it is completed and is discarded from the queue and the next process in the queue is handled by the CPU. However, if the process is not completed then it is halted (preempted) and is put at the end of the queue and then the next process as per arrival time in line is addressed during the next time slice. Round Robin reduces the penalty that short jobs suffer from FCFS by preempting running jobs periodically, and also saves starving of longer jobs and scheduling efforts in case of SJF. The main advantage of Round Robin Scheduling is that every process gets the CPU and thus there is no starvation.

The drawback of this method is that it slows down the short processes because they have to share the CPU time with other processes instead of just finishing up quickly. Thus the critical issue with the RR policy is the length of the quantum. In case it is too short, then the CPU will be spending more time on context switching or if too long then processes demanding less CPU time will suffer.





Lottery Scheduling

Lottery scheduling is a probabilistic scheduling algorithm for processes in an operating system. Processes are each assigned a number of lottery tickets, and the scheduler draws a random ticket to select the next process to be executed. The distribution of tickets need not be uniform; granting a process more tickets provides it with a relatively higher chance of selection. This technique can be used to approximate other scheduling algorithms, such as shortest job next

and Fair-share scheduling. Lottery scheduling solves the problem of starvation.

Giving each process at least one lottery ticket guarantees that it has a non-zero probability of being selected at each scheduling operation. On average, CPU time is proportional to the number of tickets given to each job. For approximation in SJF, most tickets are assigned to short-running jobs and fewer to longer running jobs. To avoid starvation, every job gets at least one ticket. Implementations of lottery scheduling should take into consideration that there could be a large number of tickets distributed among a large pool of threads. To have an array of tickets with each ticket corresponding to a thread may be highly inefficient.

We wrote algorithms for each process in C++. Further, we plotted the graphs using the matplotlib library of python for the sake of visualisation and better comparative analysis.

ANALYSIS OF SCHEDULING ALGORITHMS

The assignment of the software, known as a scheduler, used to assign priorities in a priority queue consists of mainly, CPU utilization – to keep the CPU as busy as possible. Throughput – number of processes that complete their execution per time unit. Waiting time – the amount of time a process has been waiting in the ready queue. Response time – the amount of time a process takes from when a request was submitted until the first response is produced. An ideal scheduling algorithm is one which:

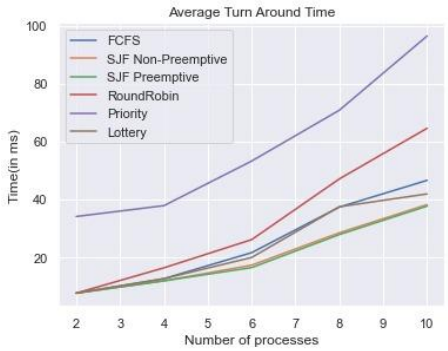
1. Minimizes Response Time which is elapsed time to do an operation (job); response time is what the user sees (e.g. time to echo keystroke in the editor, time to compile a program, real-time tasks). It must meet deadlines imposed by the World.
2. Maximizes throughput: refers to jobs per second; throughput relates to response time but is not identical. Minimizing response time will only lead to more context switching than if you maximized only throughput.
3. Minimizes overhead (context switch time) as well as efficient use of resources (CPU, disk, memory, etc.) and;
4. Maximizes fairness that is sharing CPU among users in some equitable way; not just minimizing average response time.

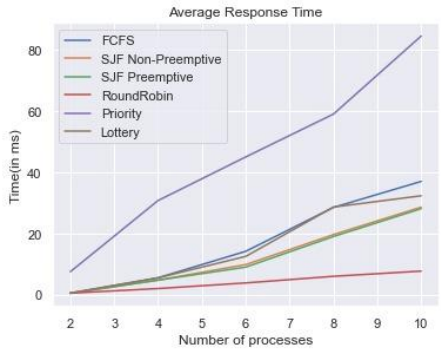
The following table shows the analysis of the different algorithms

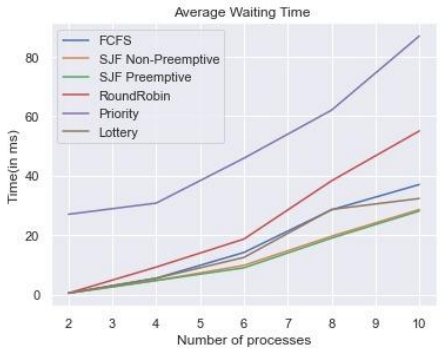
	Algorithms	FCFS	RR	SJF	SRTF	Priority	Lottery
1	CPU Utilization	Low	Medium	Higher	Requires very little overhead since it only makes a decision when a process completes or a new process is added	Higher	CPU time proportional to the number of tickets given to each job
2	Throughput	Traded off for better Response Time	Bad when the chosen quantum is small	Traded off for better Response Time	Gives the highest throughput of all scheduling algorithms.	Relatively Good	Probabilistic guarantee of throughput proportional to ticket allocation.
3	Waiting Time	The average waiting time is large	The average waiting time is large as compared to algorithms	The average waiting time is small compared to other algorithms	The average waiting time is large	Compatively On higher side	Low waiting time
4	Response Time	Good	Bad when the number of processes is large	Good	Good	Higher	Good
5	Fairness	Unfair for long jobs make short jobs wait and unimportant long jobs make important short jobs wait.	Fair	Unfair	Unfair	Fair	Fair
6	Starvation	No potential for starvation	Starvation Free	Long running CPU bound jobs can starve	Has the potential for process starvation for processes	High chances of starvation.	Solve the problem of starvation
7	Predictability	More predictable	Predictable	Impossible to predict the amount of CPU time a job has left	Impossible to predict the amount of CPU time a job has left.	Highly predictable	Less predictable
8	Preemption	Non-preemptive	Preemptive	Handles preemptive and non preemptive	Preemptive	Pre-emptive	Can be preemptive or non preemptive.

VISUAL COMPARATIVE ANALYSIS OF SCHEDULING ALGORITHMS

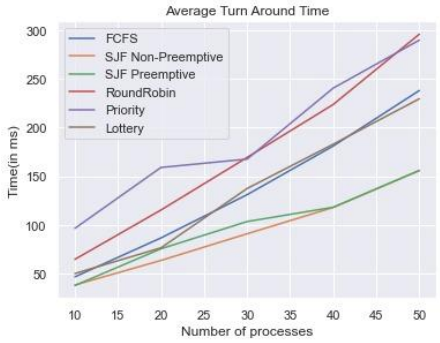
For the number of processes varying from 2-10:

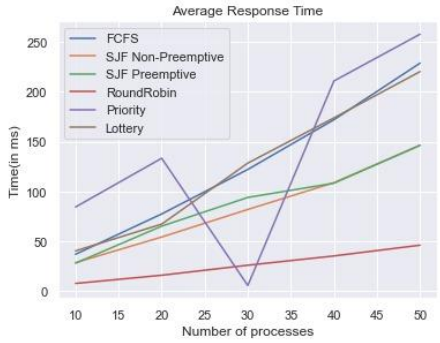


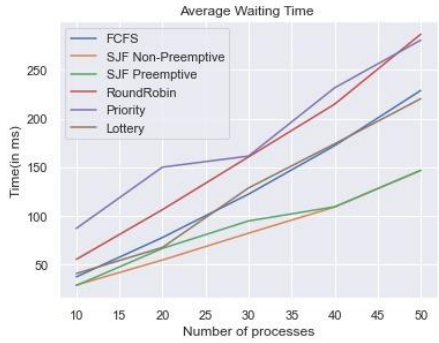




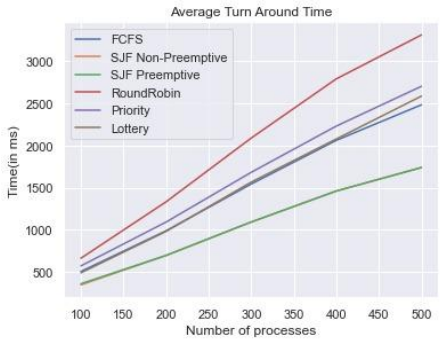
For the number of processes varying from 10-50:

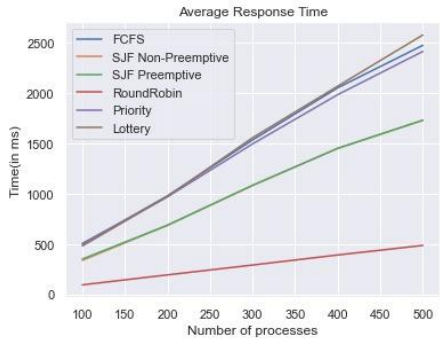


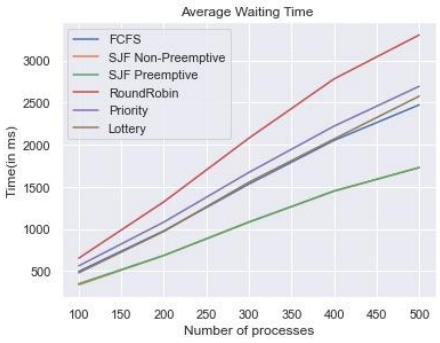




For the number of processes varying from 100-500:







~Conclusion~

For a low and moderate number of processes, priority scheduling has the highest ATAT, ART and AWT and SJT(preemptive and non-preemptive) has lowest ATAT and AWT and moderate ART, Round Robin scheduling has

Lowest ART and moderate ATAT and AWT, Lottery has a moderate ART, ATAT and AWT. For a higher number of processes, priority scheduling has moderate average times, round-robin has the highest ATAT and AWT and the lowest ART and lottery scheduling has moderate average times.

References:

GeeksforGeeks: https://www.geeksforgeeks.org/
Tutorials point: https://www.tutorialspoint.com/