Candidate Number: 249763

Effect of changing parameters on the performance CPU Scheduling Algorithms: Searching for the optimum CPU Scheduler

Introduction:

CPU scheduling involves deciding which process will be executed next in the CPU, while there are other processes on hold, ready to be completed too. There are several CPU Scheduling Algorithms that determine which process is the chosen one, the ones I will be conducting experiments on are Round Robin (RR), Ideal Shortest Job First (IdealSJF), Multi-level feedback queue with Round Robin (FeebackRR), Shortest Job First using exponential averaging (SJF), and First Come First Serve (FCFS) scheduler. Each scheduler will undergo three different scenarios, however, the overall scope of the experiments is to see how changing the parameters affect a scheduling algorithm. The first experiment will be looking at how increasing the number of processes affects the waiting time of processes depending on the scheduling algorithm. My hypothesis for this experiment is that the waiting time will decrease for IdealSJF and SJF, but increase for RR, FeebackRR, and FCFS. The second experiment will look at how increasing initial burst time estimates affect the overall turnaround time for processes depending on the scheduling algorithm. My hypothesis for the second experiment is that the turnaround time for all algorithms will increase. The last experiment I will conduct is looking at the effect of the turnaround time on the processes of each scheduler if the CPU burst time is increased. My hypothesis for the third experiment is that there will be an increase in turnaround times as the CPU burst time increases on all schedulers. CPU scheduling must be able to schedule processes efficiently, maximise CPU utilization and throughput, and minimise turnaround time, waiting time, and response time. These three experiments will be thoroughly analysed and explained to determine which scheduling algorithm to use depending on the scenarios they are involved in.

Methodology:

Experiment 1:

Scope, Objectives, and Variables:

The main objective of this experiment is to work out which scheduling algorithm to use if there are a large number of processes. To achieve this each scheduling algorithm took three different samples of the total number of processes. The three different samples were divided into three different and randomised seeds. Randomised seeds were used to allow reproducible results to be obtained so that anyone running the code for the experiment can get the same outputs. The three seeds will each have their own set of Scheduling algorithms working on a different number of processes in each seed, for instance, seed 1 will have 50 processes which each scheduling algorithm will have to work with, seed 2 with 100 seeds, and seed 3 with 200 seeds. The independent variable of this experiment is going to be the number of processes. As we are investigating the effect on the performance of algorithms we will make the dependent variable the waiting time of a process. The control variables will be all the parameters that will not be changed which tend to impact the waiting time, for example, the CPU Burst times will remain the same throughout the whole experiment.

Overall Setup:

The structure of the experiment is having a folder named experiment 1, which contains 3 seeds (seed1, seed2, seed3). Each seed will have its own input parameters and simulator

parameters for each scheduling algorithm. Output files containing results of the performances carried out by the algorithms will also be located in the corresponding seed.

Experiment 1 Seed 1: Parameters

Figure 1 below will outline the input parameters used for the input generator in seed 1. The parameters which I changed were the number of processes and the seed number. The first seed will have 50 processes as I personally found this number of processes to be the most optimum to see a change in values between the output files of each algorithm. Between all three seeds, I have kept the staticPriotity, meanInterArrival, meanCpuBurst, meanIOBurst, and meanNumberBursts the same as they are the control variables.

```
numberOfProcesses=50
staticPriority=0
meanInterArrival=150.0
meanCpuBurst=15.0
meanIOBurst=15.0
meanNumberBursts=2.0
seed=270826029269605
```

Figure 1 shows the input parameters for seed 1.

Figure 2 below shows all the simulator parameters for each scheduling algorithm, as clearly seen, all parameters are the same for each scheduler. This is because I wanted to exclude any external variable which could affect the waiting time result for each scheduler and allow the results to reflect how each algorithm operates when engaged with an increasing number of processes.

Figure 2: showing simulator parameters for each scheduler

Experiment 1 Seed 2: Parameters

Figure 3 below will show the input parameters for seed 2. To be able to observe a reasonable change in values I decided to double the number of processes (100) from the previous set of parameters. The seed number was also randomised to achieve reproducible results. However, as in the previous seed, the rest of the parameters were kept the same as the control.

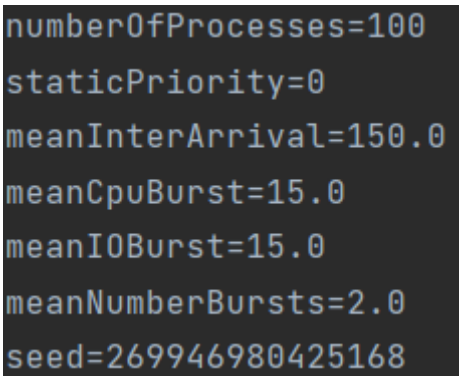

Figure 3: showing input parameters for seed 2

Figure 4 below shows the simulator parameters for each scheduler. One key parameter which was changed to each scheduler was the timeLimit for each process to be executed. Previously each scheduler had a time limit of 10000, however, the limit had to be increased to 100000, due to the fact that as you increase the number of processes each scheduler will

naturally take more time to fully complete the whole sample of processes. The rest of the parameters remained the same.

Figure 4: Shows simulator parameters for each scheduler in seed 2

Experiment 1 Seed 3: Parameters

Figure 5 below shows the input parameters for seed 3. To continue the trend of doubling sample size, I decided to experiment with the schedulers with 200 processes. Increasing the sample size by a factor of 2 allows me to be able to produce data that have a reasonable change, for increased precision leading to better conclusions. Again, a randomly generated seed was created for seed 3 and the remaining parameters were left the same.

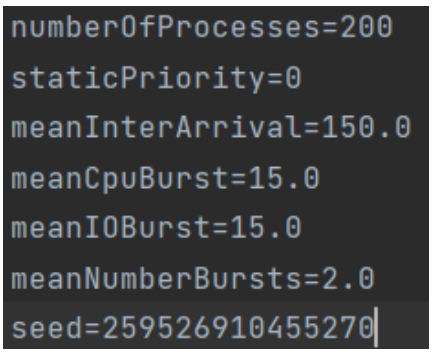

Figure 5: showing input parameters for seed 3

Figure 6 below shows the simulator parameters for seed 3. They are the same parameters as the previous seed as no parameters were needed to change to be able to investigate the change in waiting time with the increase in the number of processes.

Figure 6: showing simulator parameters for seed 3

Experiment 2:

Scope, Objectives, and Variables:

The main objective of experiment 2 is to see which scheduler would perform more efficiently if the initial burst time is increased. This means that the independent variable for this experiment is the initial burst time. This is the amount of time that a process would take to execute. Each scheduler will undergo different seeds which increase the initial burst times of the process, to see how the performance of each scheduler is affected. The dependent variable will be the turnaround time, this is the time that a process would take to be completed from when it was created. My control variables will be all the remaining parameters which will stay constant throughout the experiment.

Overall Setup:

The setup for experiment 2 will be the same as experiment 1, with 3 different seeds, input parameters, and simulator parameters for each seed.

Experiment 2 Seed 1: Parameters

Figure 7 below shows the input parameters for this experiment 2, seed 1. I have used 50 as the number of processes to be able to observe the reasonable change in values in the output files between all schedulers. I have also randomly generated the seed number for this seed for reproducible results.

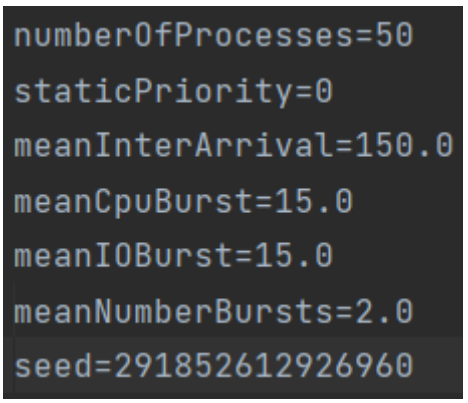

Figure 7: showing input parameters for seed 1

Figure 8 below shows the simulator parameters for each scheduler. The key change in the parameters is the initial burst time for each scheduler. The starting initial burst will be 10, this number had no influence on any other variable, it was randomly picked, as I just needed an initial time. The remaining parameters will act as a control so they will not be changed throughout the experiment.

Figure 8: showing simulator parameters for seed 1

Experiment 2 Seed 2: Parameters

Figure 9 below will show the input parameters used for seed 2, as we are only dealing with the initial burst time none of the input parameters were changed, apart from the seed number.

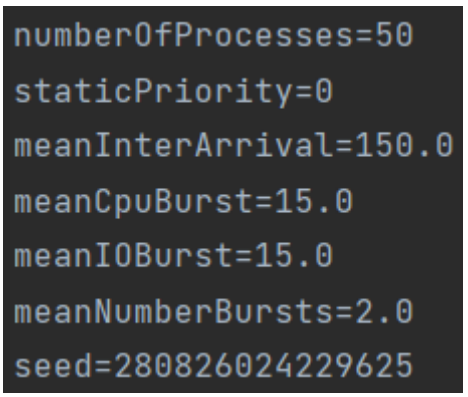

Figure 9: showing input parameters for seed 2

Figure 10 below will show the simulator parameters used for seed 2. The initial burst time has been increased by 5 to see the effect of the turnaround time. It was increased by 5 so that we could see a change in times more visibly for precise results. The remaining parameters were left the same.

Figure 10: showing simulator parameters for seed 2

Experiment 2 Seed 3: Parameters

Figure 11 below shows the input parameters for seed 3, as we are only dealing with the initial burst time none of the input parameters were changed, apart from the seed number.

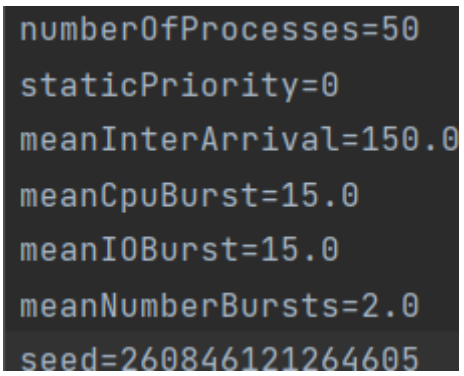

Figure 11: shows input parameters for seed 3

Figure 12 below will show the simulator parameters used for seed 3. The initial burst time has been increased by 5 again to see the effect of the turnaround time. It was increased by 5 again to see if there have been any changes and also to only increase the burst time in a linear fashion. The remaining parameters were left the same.

Figure 12: shows the simulator parameters for seed 3

Experiment 3:

Scope, Objectives, and Variables:

The main objective of this experiment is to investigate the turnaround time of each scheduler when the CPU burst time is increased. This experiment is looking at which scheduler you would want if put in a scenario, where each process needs an increasing amount of time in the CPU. This means that my independent variable is the change in CPU burst times, and my dependent variable is the turnaround time of each scheduler. The control variables would be all parameters that were left as default and constant throughout the experiment. I have also decided to keep seed numbers the same to observe how it could affect the overall performance of each scheduler. The CPU burst time is the amount of time that a process requires in the CPU in order to be executed. When increasing this time, schedulers would take longer to fully complete all processes. Thereby I want to observe the difference between schedulers in terms of how long it takes for the whole process to be terminated(Turnaround time).

Overall Setup

The setup for experiment 3 will be the same as experiments 1 and 2, with 3 different seeds, input parameters, and simulator parameters for each seed. However, I will not be changing the seed numbers, the seeds will act as trials.

Experiment 3 Seed 1: Parameters

Figure below shows the input parameters of seed 1 of experiment 3. I have kept their default values so that I can use seed 1 as a base.

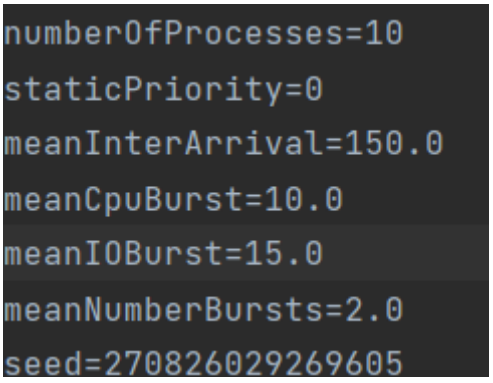

Figure 13: shows the simulator parameters for seed 1

Figure 14 below will show the simulator parameters used for seed 1. All Schedulers have the same parameters. I left the constant to make them control variables, so no external variables can affect the results.

Figure 14: shows simulator parameters for seed 1

Experiment 3 Seed 2: Parameters

Figure below shows the input parameters for seed 2 of experiment 3. I have now adjusted the mean CPU burst time (increased by 5 ms). The remaining parameters are the same as the previous seed.

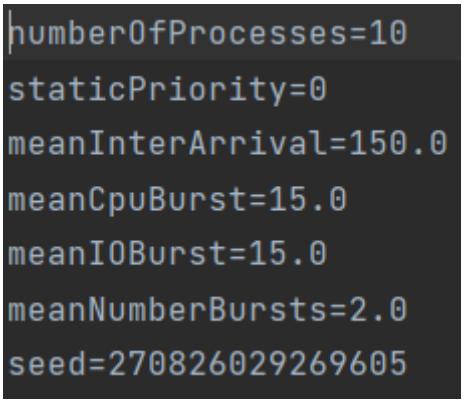

Figure 15: shows the simulator parameters for seed 2

Figure 16 below will show the simulator parameters used for seed 2. All Schedulers have the same parameters. I left the constant to make them control variables, so no external variables can affect the results.

Figure 16: shows simulator parameters for seed 2

Experiment 3 Seed 3: Parameters

Figure below shows the input parameters for seed 3 of experiment 3. I have now adjusted the mean CPU burst time (increased by 5 ms). The remaining parameters are the same as the previous seed.

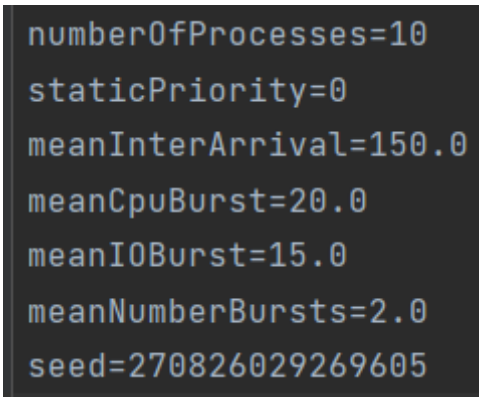

Figure 17: shows the simulator parameters for seed 3

Figure 18 below will show the simulator parameters used for seed 2. All Schedulers have the same parameters. I left the constant to make them control variables, so no external variables can affect the results.

Figure 18: shows simulator parameters for seed 3

Results:

Experiment 1, 2, and 3 Results:

These bar charts were generated using Excel. I took the output files and imported them. After importing them I calculated the average waiting time for each scheduler and produced a table, with the Average waiting time on the y-axis and I used the x-axis to layout all schedulers. I decided to use bar chartst as I believe it was the most suitable for representing this data, it allows anyone to clearly see which scheduler is most efficient straight away by looking for the shortest bar. I also left the corresponding average waiting time on the bar for convenience.

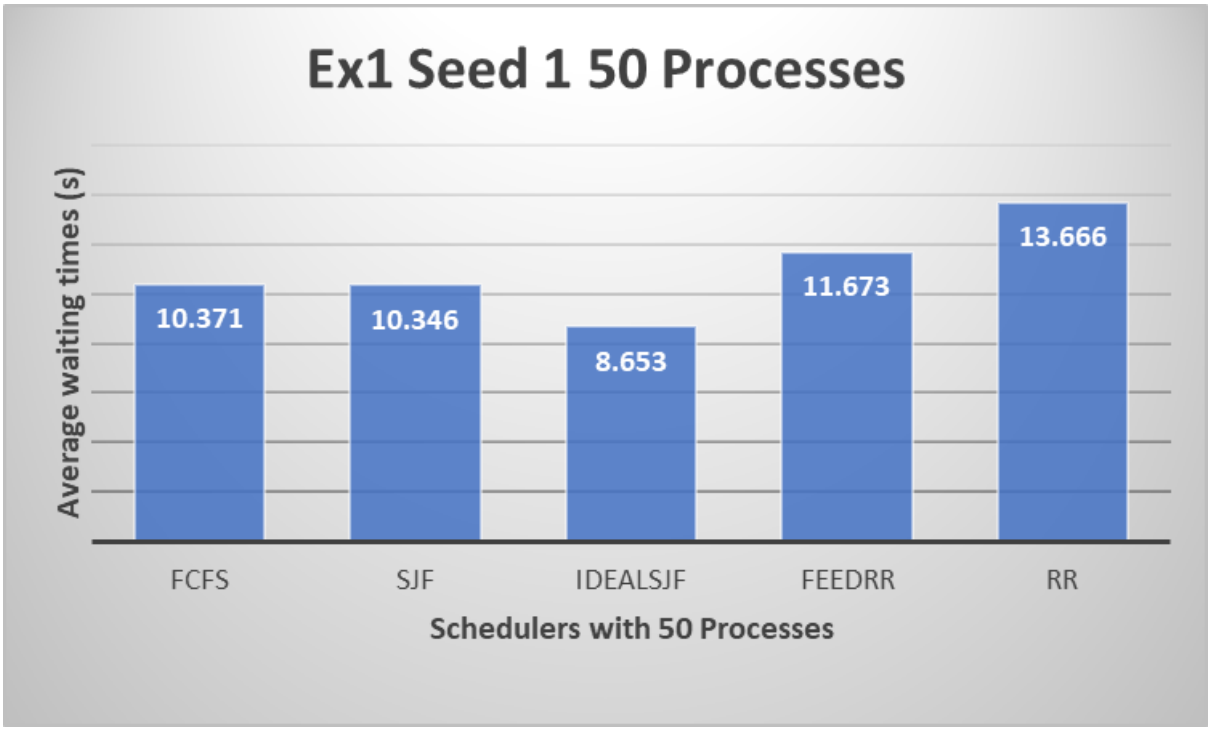

Figure 19: shows a bar chart of experiment 1 seed 1

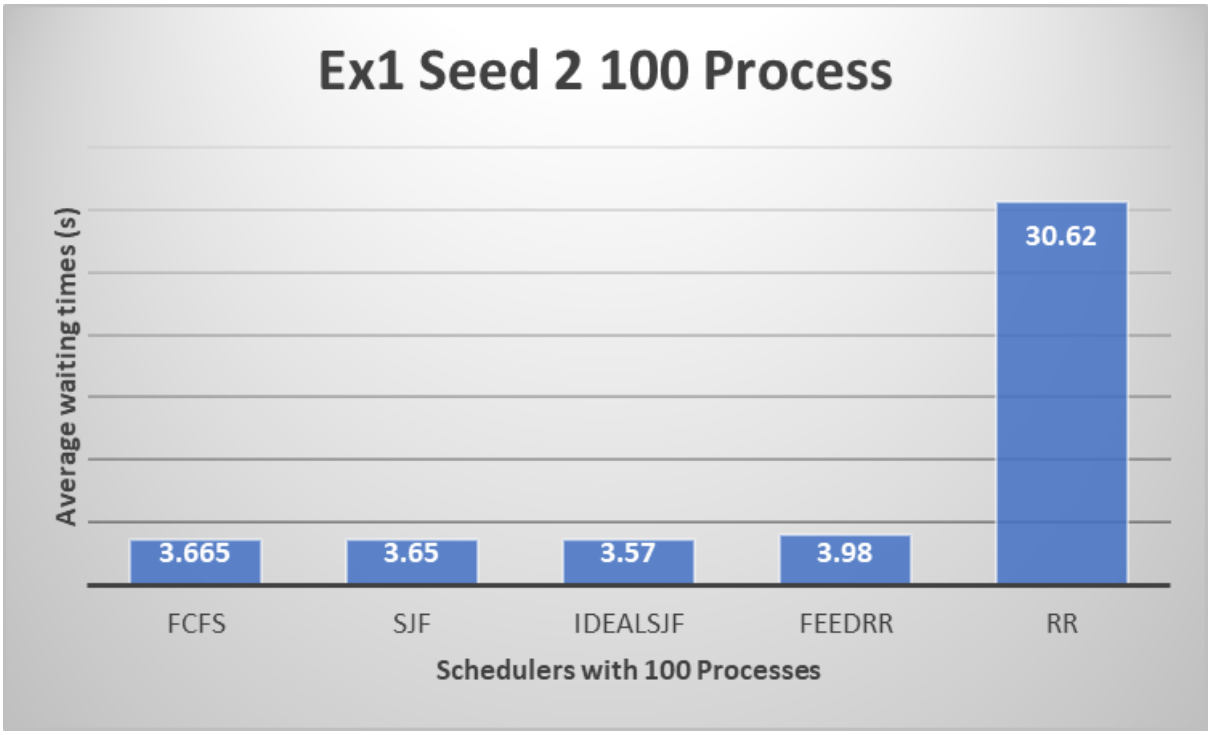

Figure 20: shows a bar chart of experiment 1 seed 2

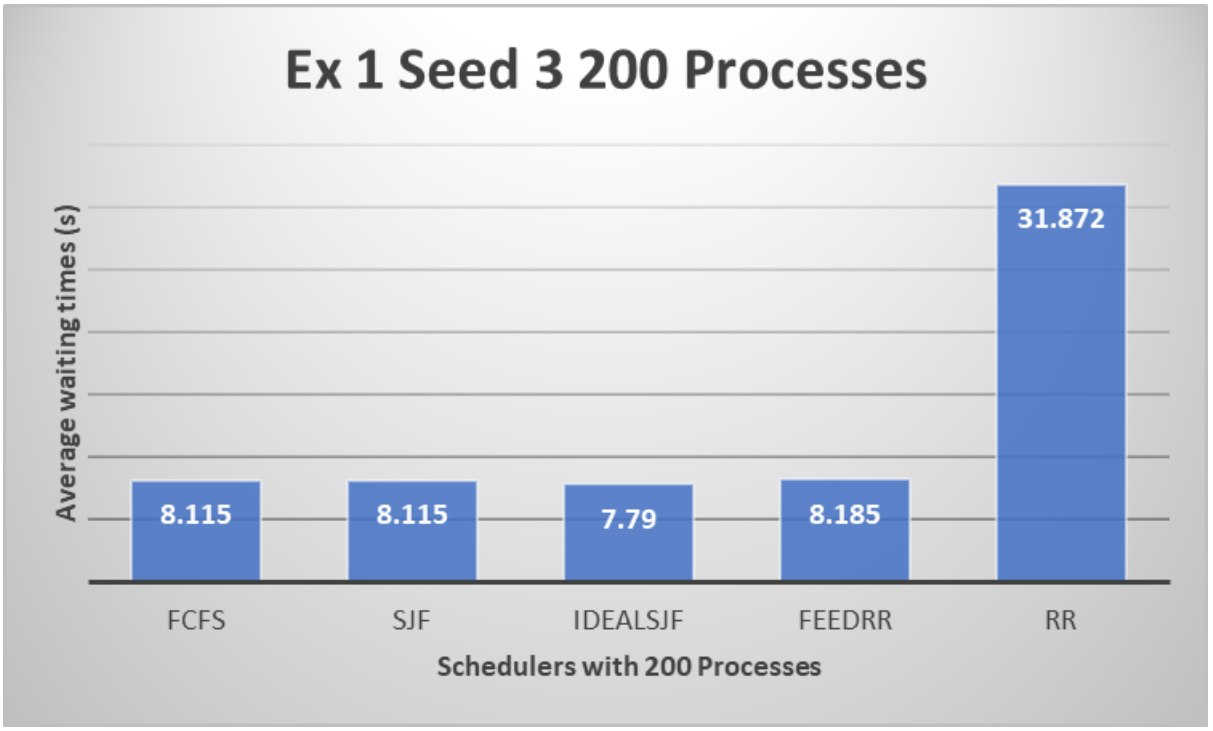

Figure 21: shows a bar chart of experiment 1 seed 3

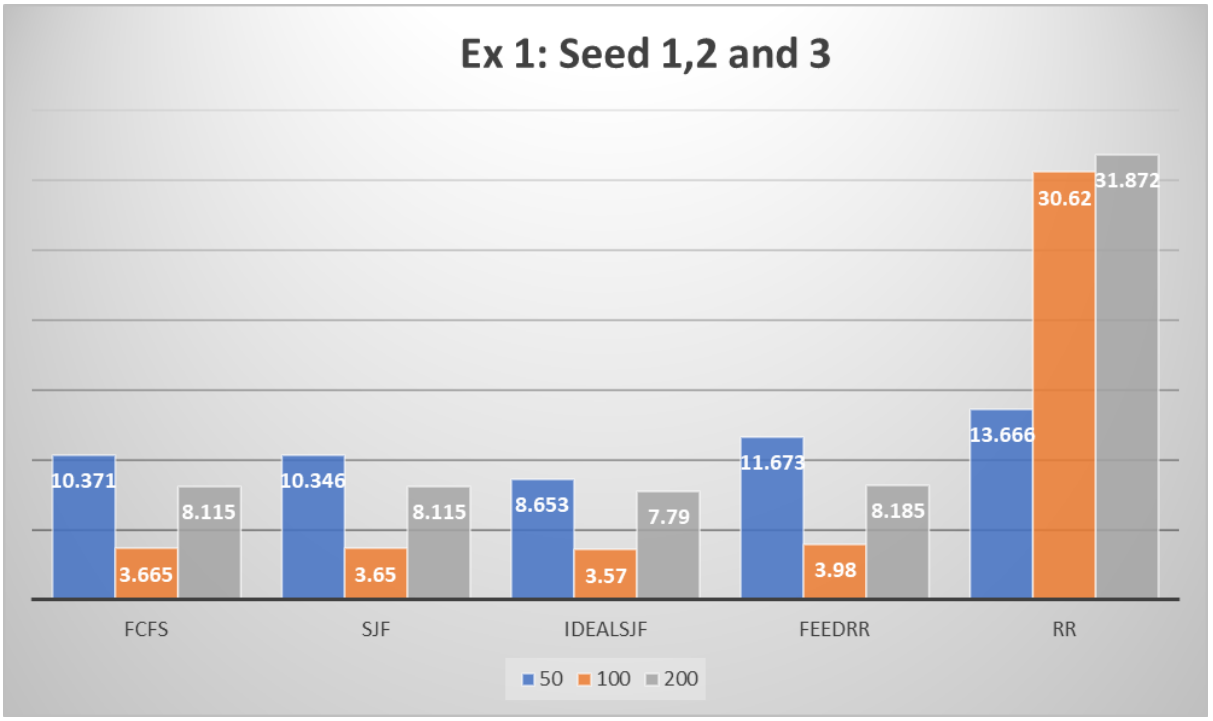

Figure 22: shows a combined bar chart of experiment 1 seed 1, 2 and 3.

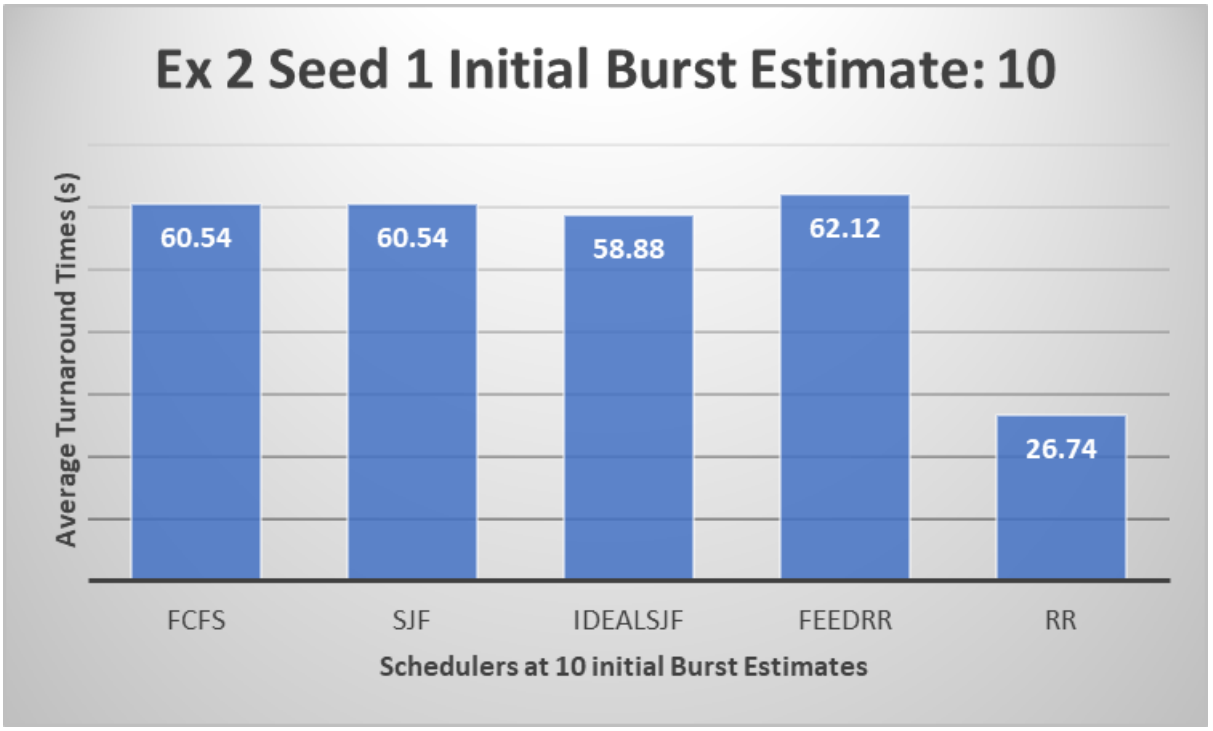

Figure 23: shows a bar chart of experiment 2 seed 1

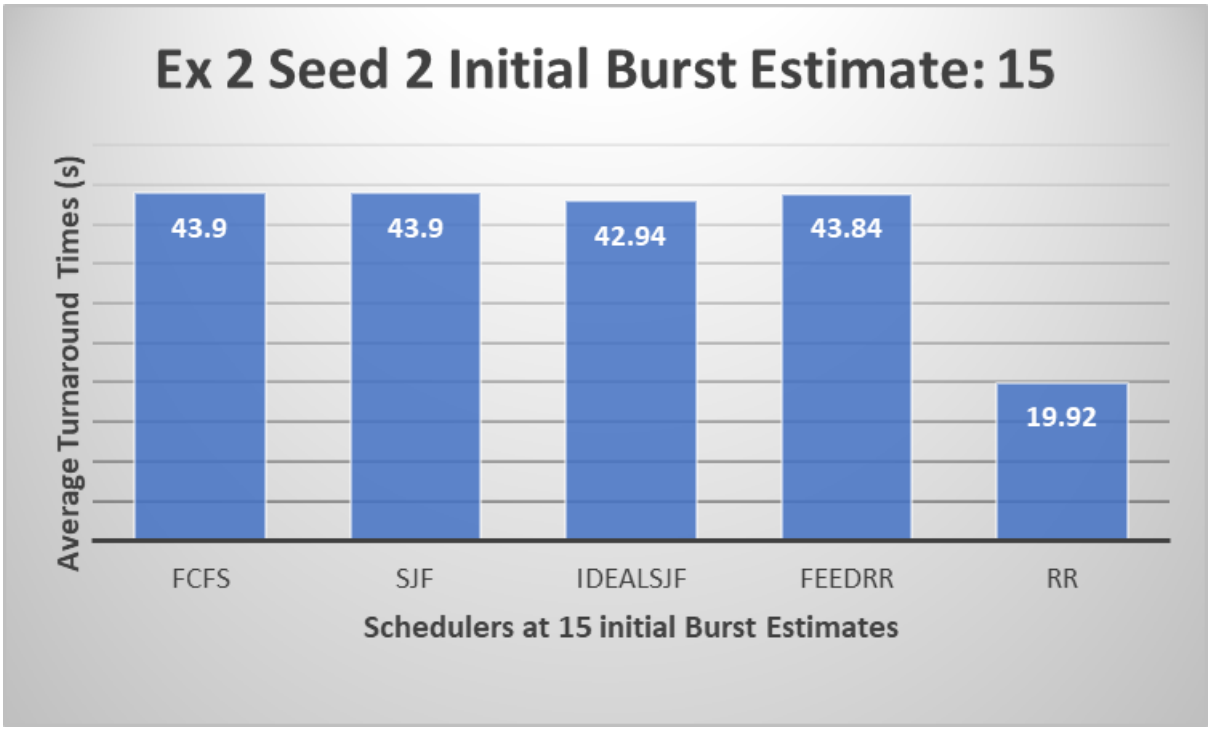

Figure 24: shows a bar chart of experiment 2 seed 2

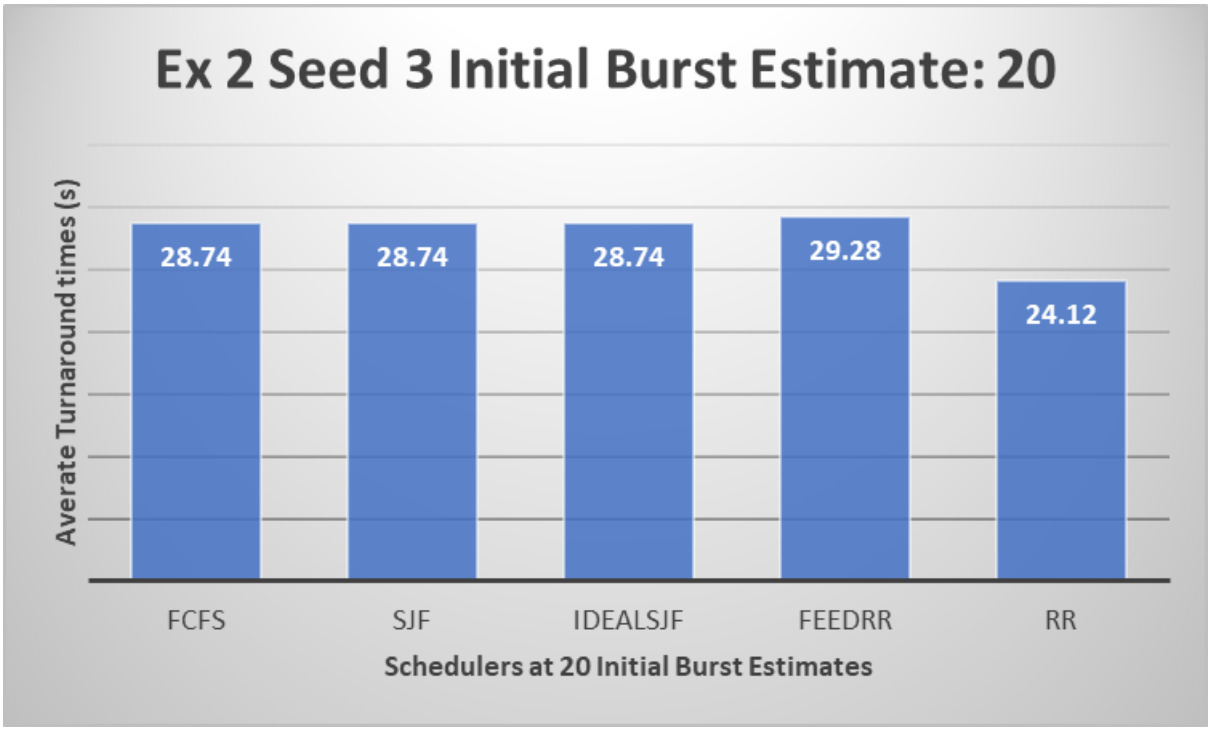

Figure 25: shows a bar chart of experiment 2 seed 3

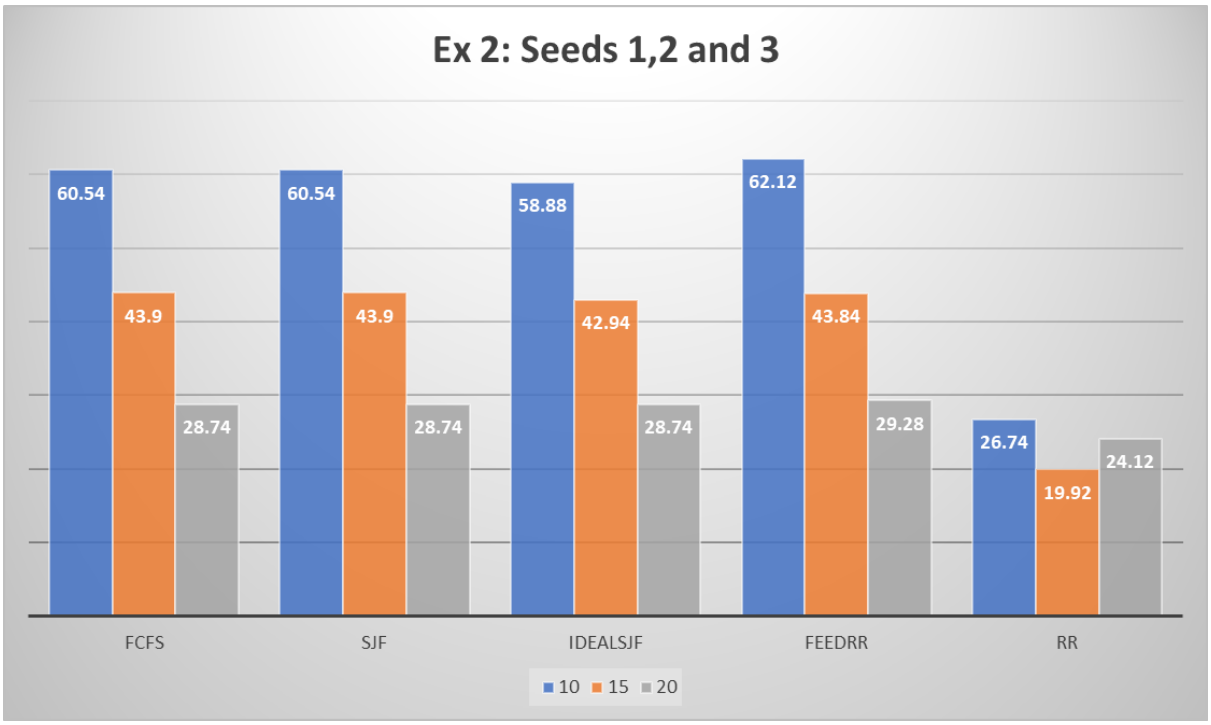

Figure 26: shows a combined bar chart of experiment 2 seed 1, 2, 3

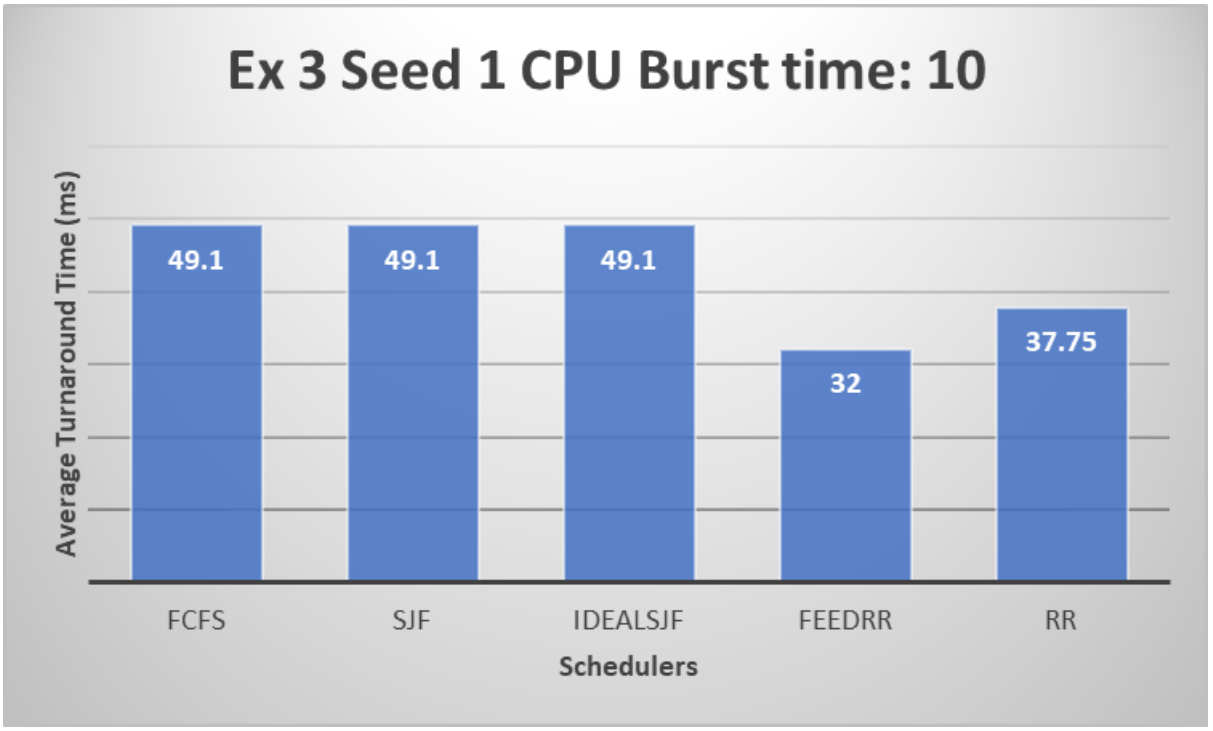

Figure 27: shows a bar chart of experiment 3 seed 1

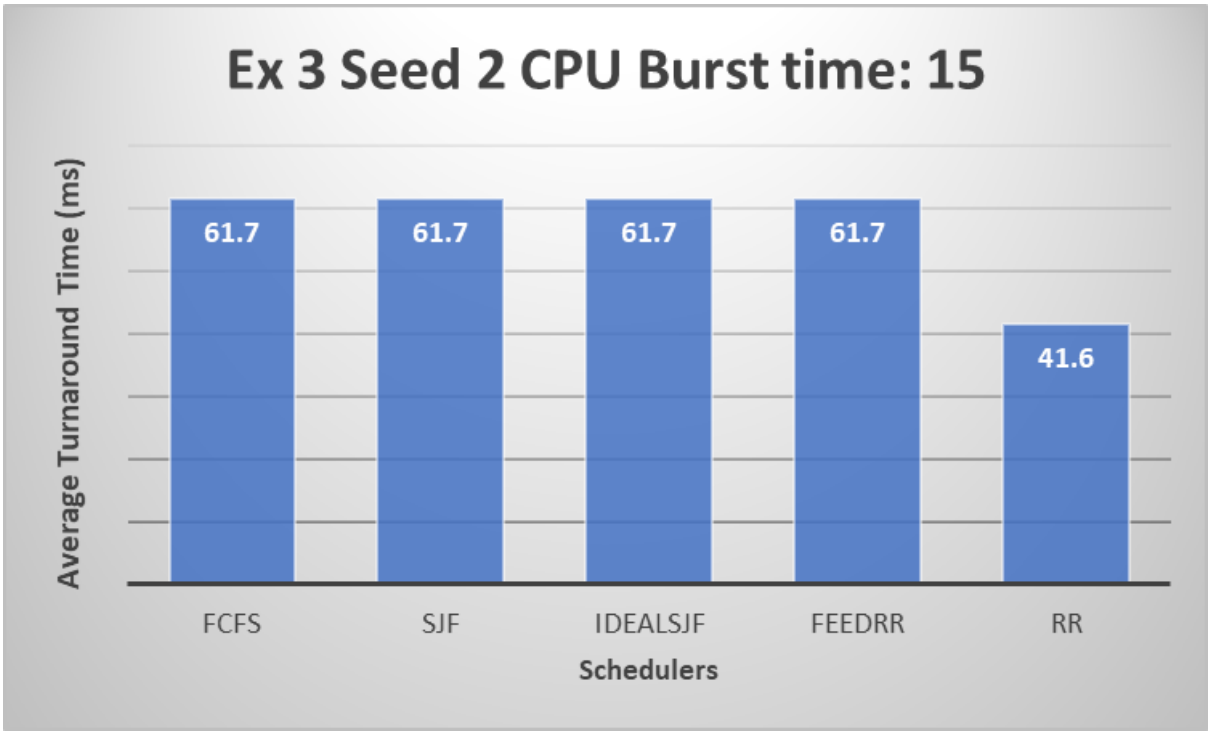

Figure 28: shows a bar chart of experiment 3 seed 2

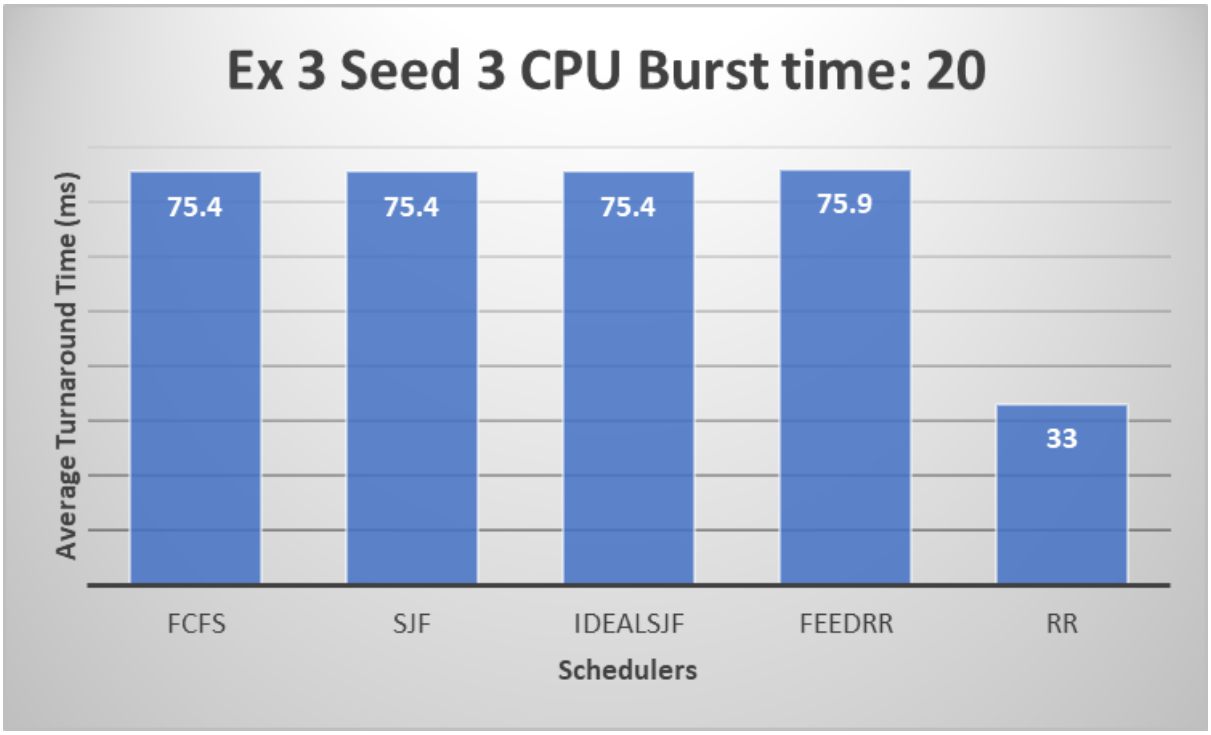

Figure 29: shows a bar chart of experiment 3 seed 3

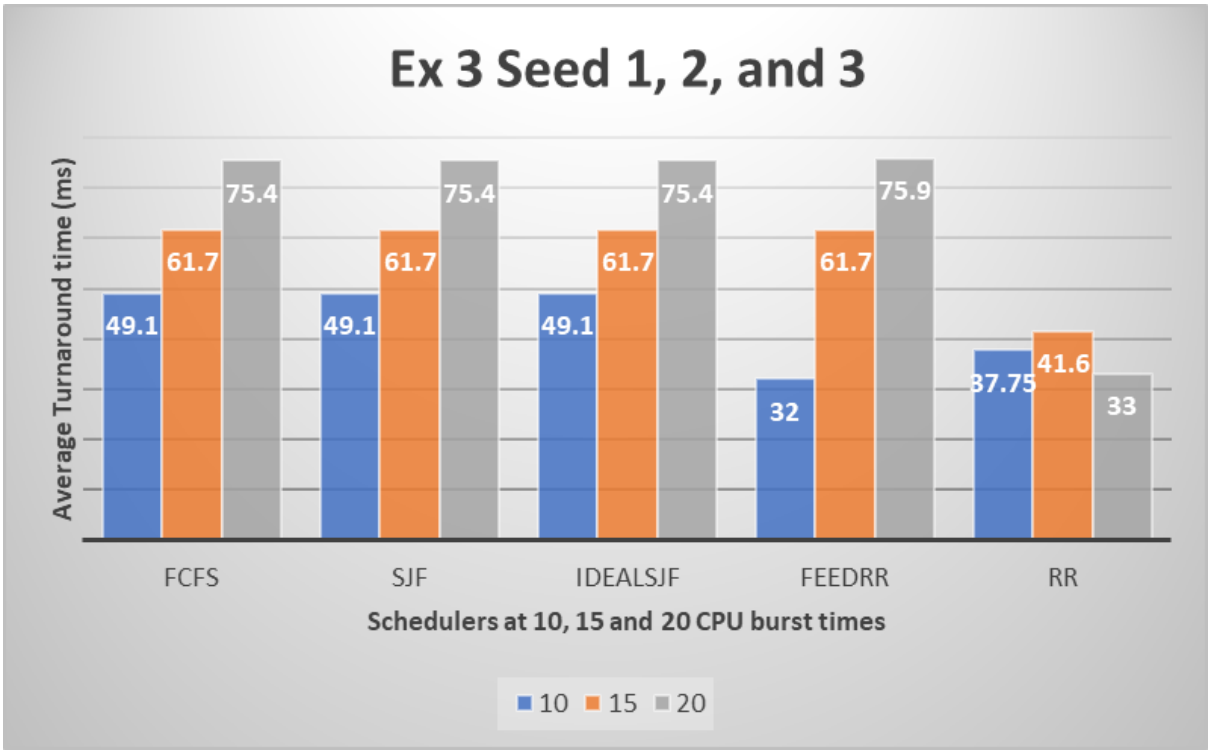

Figure 30: shows a combined bar chart of experiment 3 seed 1, 2, 3

Discussion:

Experiment 1:

My hypothesis for this experiment was that the waiting time will decrease for IdealSJF and SJF, but increase for RR, FeebackRR, and FCFS. I reject my hypothesis as I believe that I do not have sufficient evidence to support my hypothesis. To explain myself I will look at each scheduler individually and analyse its performance. FCFS, First come first serve algorithm works in a way in which a process that has arrived is put in the ready queue instantly and executed in the order they arrived at. In theory, if there are an increasing amount of processes that are arriving, the length of the ready queue, naturally increases, meaning that processes would have to wait longer, hence why my hypothesis stated that the waiting time would increase. However, from seed 2 to seed 3, the waiting time did increase, but this is still not enough information to support my hypothesis. I would need to carry out more trials in this experiment to see a clear trend and relationship between the scheduler and waiting time. SJF, the Shortest job first algorithm works by assigning a higher priority to the shortest burst time process, which means that the higher the priority the sooner the process is taken from the ready queue and executed. I predicted that the average wait time would decrease which from the results table overall it has. However, I still have to reject my hypothesis, as between seeds 2 and 3, there was an increase in waiting time. My reason to believe that the waiting time would have an overall decrease was that there would be more processes with lower burst times meaning that the processes would be terminated sooner, allowing other processes to be executed in less time. Ideal SJF, Ideal shortest job first was also predicted to have a decrease in waiting time. The scheduler works the same way as SJF, but it knows which processes are involved in advance. The explanations for SJF apply to Ideal SJF in this experiment too. However, one significant point I can draw out from the three charts is that Ideal SJF has consistently the lowest average waiting time. Feedback RR uses two queues to set processes with priority levels based on their burst times and then uses Round Robin to work out which process is executed, From the graphs, we can see the same trend as FCFS, a decrease and then an increase in waiting time, with an overall decrease. This fluctuation is not enough to support my hypothesis. RR, Round Robin, uses time quantum which is a set timer in which a process spends executing, and then uses an FCFS system to determine who goes next, allowing a fair share of processing time. My hypothesis has correctly predicted that the waiting time for this scheduler would increase, this is due to the fact that there would be more processes waiting on the ready queue, with the likelihood of having high burst times. However, there is an issue with this graph and its data for RR. When looking at the table of results. I could see that only 54 processes were detected by Excel even though my output file had all processes.

Experiment 2:

My hypothesis for this experiment was that the turnaround time for all algorithms will increase. This is due to the fact that if the initial burst time is increased, the longer each process will take time to be executed, increasing the difference between each process' created time and terminated time, which equals the turnaround time. However, from figure 20, we can see that as the value of the initial burst time increases, the average turnaround time is decreasing. This is the case for all scheduling algorithms, making my results unable

to support my hypothesis. However, I am not satisfied with the results as in theory, my results should have produced an overall increasing trend, this is because all scheduling algorithms will have a longer turnaround time due to the increasing size of the process in terms of time taken. From figure 20, I can derive that the RR scheduler is an anomaly, this is due to the fact that not all 50 processes were involved so it does not represent the whole sample. However, if you look at the other four schedulers we can see that IdealSJF is the optimal scheduling algorithm when one has large values for initial burst times, the bar charts in Figures 17, 18, and 19, the turnaround time is relatively lower than other schedulers, making it most efficient.

Experiment 3:

My hypothesis for this experiment was that there will be an increase in turnaround times as the CPU burst time increases on all schedulers. Figure 30 shows that each scheduler has increasing turnaround times on average. The reason for this increase is that if the CPU burst times increase the more time the processes require in the CPU to be executed, therefore increasing the turnaround time as this is the value that measures how long a process takes to be completed from when it started executing. However, figure 30, also shows that the RR scheduler fluctuated in the turnaround time, it had an increase and then a decrease in time. This goes against my hypothesis. I believe the reason for this is because the results for the output file for RR in seed 2 and 3, contained fewer processes, even though its input files have the same parameters for all schedulers. I think the issue might be coming from the time limit of the RR scheduler, it forces the scheduler to end before scheduling all processes. Therefore, I cannot conclude that my hypothesis is fully supported by the results, as they are not complete and accurate, It would be biased to compare the schedulers and work out which one is most efficient if the whole sample size was not taken into account. If we exclude the RR scheduler from figure 30 and focus on which scheduler is performing better, we can see that Feedback RR is constantly providing smaller average turnaround times across all schedulers. This could be due to the fact of the characteristics and policies of the algorithm. Feedback RR makes an assumption of the burst times of processes in advance so that it can then use RR to efficiently schedule the next highest priority process, this can save lots of time in terms of processes waiting, meaning more processes can be completed sooner, decreasing turnaround times.

Threats to validity:

In almost all experiments I had a sense that some factors were a massive threat to my validity and how precise/ accurate my results would be. First of all, in terms of the back end, I was not fully confident in my working schedulers as well as my process methods. I had some doubts about my implementation of all the methods. This is a threat to my validity as if my schedulers are not functioning as they should be, they will not produce the correct results. Another threat to my validity was the importation of output files into Excel, I could clearly see many processes and their data was lost. In the output files, I would have all 100 processes for example, and when I transferred them to Excel for analysis and chart generations. I would only have 54 processes' worth of data. After many attempts and using alternative software, such as, Google Sheets, I was unable to produce the accurate results I was looking for.

Conclusions:

Experiment 1:

Experiment 1 looked at which scheduler would work best for an increasing number of processes, the metric used to make comparisons was the waiting time of each scheduler, after running the experiment and visually observing the results, I cannot conclude that the experiment was successful at depicting which scheduler was best, as it did not support my hypothesis and there were some missing data. However, Feedback RR is an honorable mention, as it did have the lowest waiting time on average when the number of processes was increased across all schedulers.

Experiment 2:

Experiment 2 investigated which scheduler would perform better if the initial burst time was increased for processes, meaning processes take longer to be executed. Each scheduler took the same number of processes and produced unrealistic results. It was expected that average turnaround times would increase due to longer burst times, however, the results show a decrease across all schedulers. As, there were a few threats to the validity and accuracy of this data including, the time limit is set too low. We cannot conclude which scheduler is best for this scenario. Honorable mention for this experiment does go to IdealSJF, as it provided the lowest turnaround times across all schedulers.

Experiment 3:

Experiment 3 searched for the best scheduler to use if the CPU Burst time was increased. It was the only experiment that partially supported my hypothesis. However, due to the anomaly and inaccuracy of the RR scheduler, I cannot conclude which scheduler was best at dealing with an increased CPU burst time. As it would be biased to exclude RR from the equation.

In conclusion, my experiments were not very successful at exactly finding out the optimum scheduler for a certain scenario, but there are hints in some results which can be used to determine the best scheduling algorithm to use for efficient scheduling.