Selahattin Cem Öztürk 21802856
CS342-2 Project 2

First, I run the simulation for the three scheduling algorithm with random PL and IAT's, Where the mean of the IAT was 30 and the range of it was [10,100]; and PL mean was 70 in the range of [50, 200]. I generated 20 PL values by using these. IAT value chosen smaller than PL values intentionally, otherwise CPU's finish the process and starts to waiting for main thread to wake up and send another PL, results are shown below;

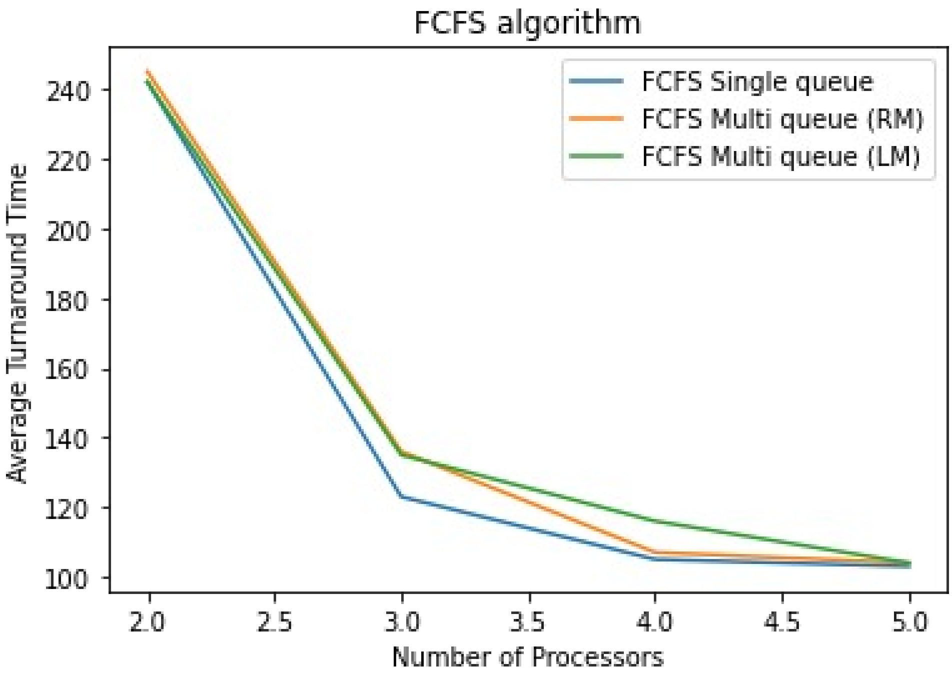


*Fig. 1: FCFS algorithm's performance with different CPU counts and queue models.*

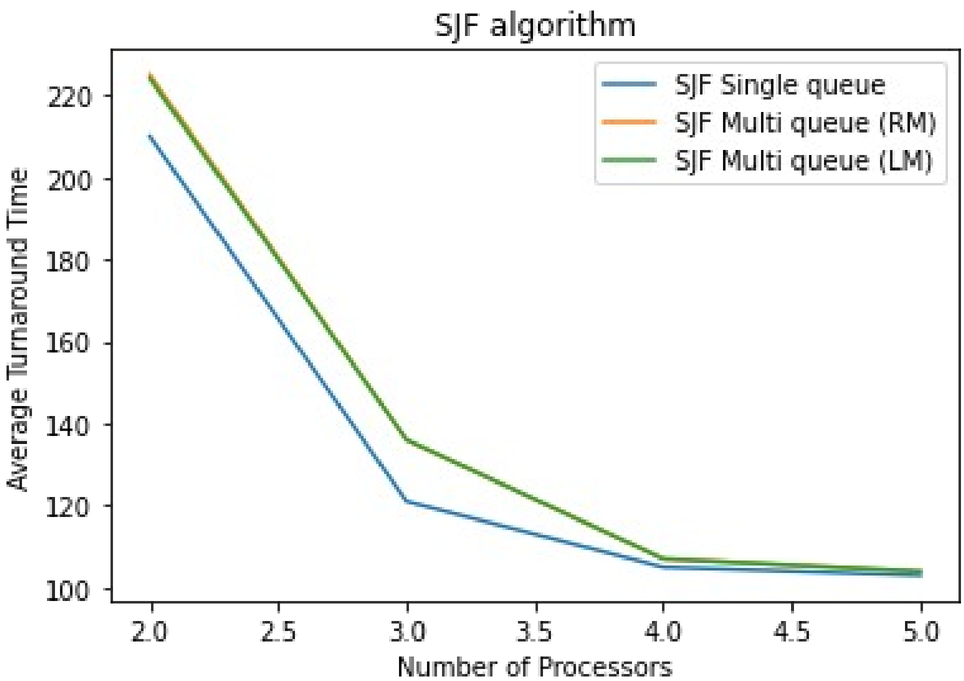


*Fig. 3: SJF algorithm's performance with different CPU counts and queue models.*

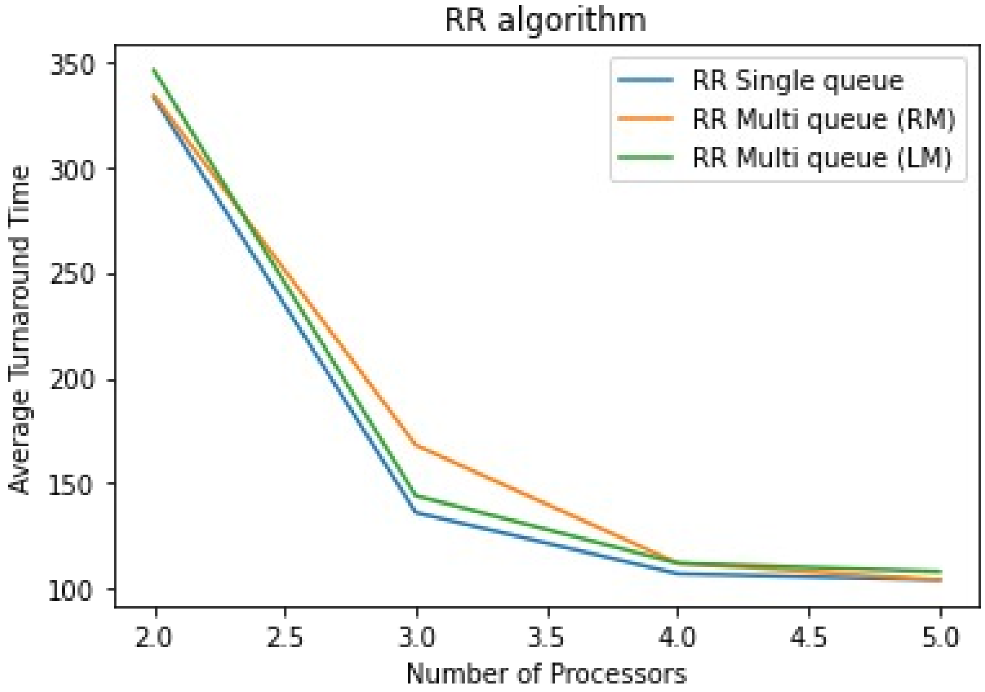


*Fig. 3: RR algorithm's performance with different CPU counts and queue models.*

As seen from the graphs, SJF algorithm is better than the other two. However, in real life we do not know the real burst length of processes. Thus, although it is better algorithm, in real life it needs a expected burst length formulation inside the kernel.

If we compare the different queue approaches, single queue is better; but this is probably due to my simulation values. Another reason might be some queues stays empty. For example main thread put some processes in queue 1 and 2 but when it is going to put another process in some queue, if the processes done in queue 1, it might put it into queue 1 again. I this case process 3 does not utilized. This problem occurs in load balancing method since if there is more than one smallest total burst length, it chose the smaller queue id. In the RM, another problem occurs if the processes are to short. For example it puts a process that is to short in the cpu 3. When cpu 3 finishes it, it does not do any job until another queue is putted in the queue. Unfortunately comparison of RM and LM method is not possible from these graphs, In the next graph it will become more clear.

I also analyze different Q values for RR algorithm. Since if Q value is way to big, it turns into FCFS algorithm, I chose the Q values in the interval [0,70] (70 is the mean value of my PL's). I also increase the PL count to 40 and used 4 processors. Results are given below.

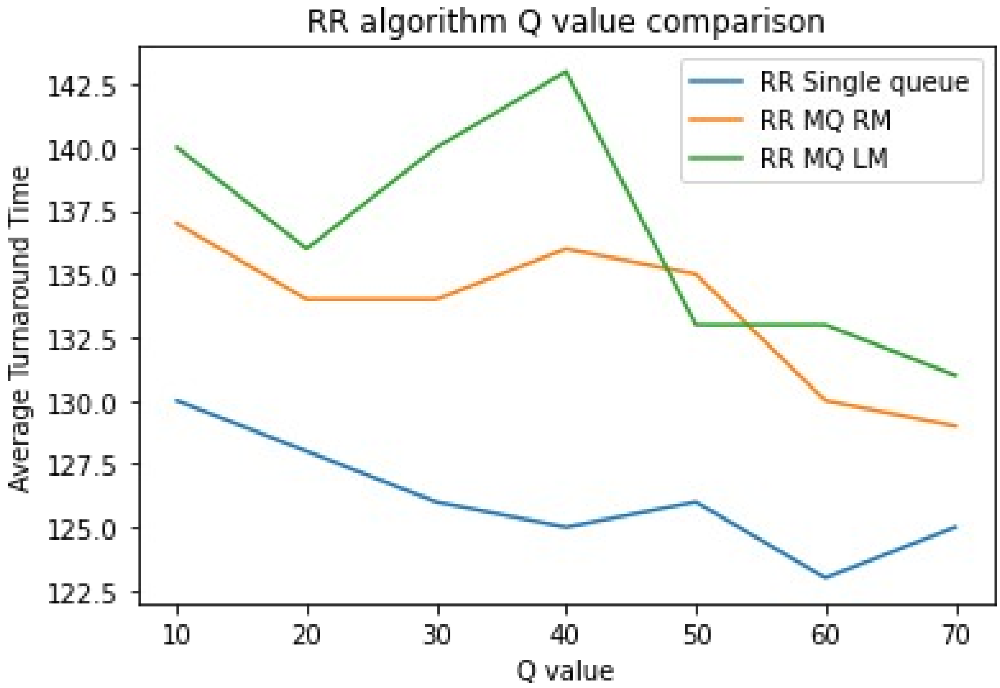


*Fig. 4: RR algorithm's performance with different Q values and queue models. (N=4)*

In this table we can clearly see that single queue approach is better, second best is RM algorithm. Also as q value increases, the results generally gets better. But we know that increasing Q too much turn RR algorithm into FCFS algorithm. For example, simulation runned with the following parameters;

./mps -n 2 -a M RM -s RR 400 -m 1 -r 0 0 0 100 50 400 30

./mps -n 2 -a M RM -s FCFS 0 -m 1 -r 0 0 0 100 50 400 30.

Results show that RR algorithm's performance is same with FCFS. This is because I chose a Q value that is the maximum of the burst time which varies between 50 and 400. Simulation results are given below.

```
(base) cem@cem-GF63-Thin-9SC:~/Desktop/cs342_project2$ ./mps -n 2 -a M RM -s FCFS 0 -i infile.txt -m 1 -r 0 0 0 100 50 400 30
Simulation started with values:
n = 2, SAP = M, QS = RM, ALG = FCFS, Q = 0, infile = infile.txt, outmode = 1, outfile = out.txt, T = 0, T1 = 0, T2 = 0, L = 100, L1 = 50, L2 = 400, PC = 30


pid     cpu     burstlen        arv     finish  waitingtime     turnaround
1       1       54              0       55      1               55
2       2       65              0       66      1               66
3       1       244             0       299     55              299
4       2       307             0       373     66              373
5       1       154             0       454     300             454
6       2       220             0       594     374             594
7       1       99              0       553     454             553
8       2       107             0       701     594             701
9       1       359             0       912     553             912
10      2       61              0       762     701             762
11      1       195             1       1107    911             1106
12      2       139             1       901     761             900
13      1       105             1       1212    1106            1211
14      2       98              1       999     900             998
15      1       146             1       1358    1211            1357
16      2       288             1       1287    998             1286
17      1       133             1       1491    1357            1490
18      2       195             1       1483    1287            1482
19      1       300             1       1792    1491            1791
20      2       60              1       1543    1482            1542
21      1       133             1       1925    1791            1924
22      2       91              1       1634    1542            1633
23      1       233             1       2158    1924            2157
24      2       163             1       1797    1633            1796
25      1       367             2       2525    2156            2523
26      2       52              2       1849    1795            1847
27      1       128             2       2653    2523            2651
28      2       70              2       1919    1847            1917
29      1       90              2       2743    2651            2741
30      2       173             2       2093    1918            2091
Average turnaround time: 1307 ms(base) cem@cem-GF63-Thin-9SC:~/Desktop/cs342_project2$ 
```

*Fig. 5: Simulation result of RR algorithm.*

```
(base) cem@cem-GF63-Thin-9SC:~/Desktop/cs342_project2$ ./mps -n 2 -a M RM -s RR 400 -i infile.txt -m 1 -r 0 0 0 100 50 400 30
Simulation started with values:
n = 2, SAP = M, QS = RM, ALG = RR, Q = 400, infile = infile.txt, outmode = 1, outfile = out.txt, T = 0, T1 = 0, T2 = 0, L = 100, L1 = 50, L2 = 400, PC = 30


pid     cpu     burstlen        arv     finish  waitingtime     turnaround
1       1       54              0       55      1               55
2       2       65              0       66      1               66
3       1       244             0       299     55              299
4       2       307             0       373     66              373
5       1       154             0       453     299             453
6       2       220             0       593     373             593
7       1       99              0       553     454             553
8       2       107             0       701     594             701
9       1       359             0       912     553             912
10      2       61              0       762     701             762
11      1       195             1       1107    911             1106
12      2       139             1       901     761             900
13      1       105             1       1212    1106            1211
14      2       98              1       999     900             998
15      1       146             1       1358    1211            1357
16      2       288             1       1287    998             1286
17      1       133             1       1491    1357            1490
18      2       195             1       1482    1286            1481
19      1       300             1       1792    1491            1791
20      2       60              1       1543    1482            1542
21      1       133             1       1925    1791            1924
22      2       91              1       1634    1542            1633
23      1       233             1       2158    1924            2157
24      2       163             1       1797    1633            1796
25      1       367             1       2525    2157            2524
26      2       52              2       1849    1795            1847
27      1       128             2       2653    2523            2651
28      2       70              2       1919    1847            1917
29      1       90              2       2743    2651            2741
30      2       173             2       2093    1918            2091
Average turnaround time: 1307 ms(base) cem@cem-GF63-Thin-9SC:~/Desktop/cs342_project2$ ./mps -n 2 -a M RM -s FCFS 0 -i infile.txt -m 1 -r 0 0 0 100 50 400 30
```

*Fig. 6: Simulation result of FCFS algorithm.*

From this result we can say that FCFS algorithm's average turnaround time is same with RR algorithm's when the Q value is the maximum burst length.