Project Report on HS2 Line from London to Birmingham

Victory Chimamaka Uwaoma
Msc. Data Analytics
National College of Ireland
Dublin, Ireland
x19210931

I. Introduction

This project talks about the travelling speed of trains on the HS2 line from London Old Oak Common station to Birmingham Interchange station along with the signaling blocks in each station. The simulation model will be verified by inducing a temporary break down of 30 minutes due to electrical malfunction and introducing variability of traveling times using log-normal distribution as described by [1].

This project hinges on the following assumptions

1. No impact of air resistance (i.e. constant acceleration).
2. Only one train in one signaling block at a time.
3. Signaling blocks have all equal length.
4. Assume the track consist of ***k*** signaling blocks and fixed number of trains ***n*** per hour.

The train is modelled as a single queue and single server system where the signaling blocks act as the servers while the trains are queued up and can only be served one at a time by a signaling block. A visual representation of the train queueing system is shown below.

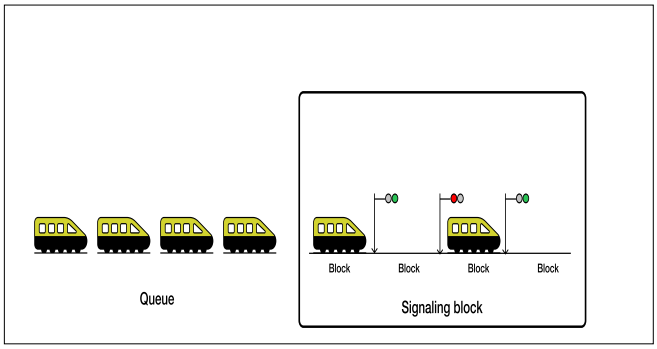

Figure 1: The train queueing system

The figure 2 below shows the output of the data worked with. It includes the line of the train, the starting point of the train, its destination, the travel time of the train as well as the assumed number of signalling blocks.

	Line	From	To	travelTime	signalingBlocks
0	HS2	London Old Oak Common	Birmingham Interchange	1684	14

Figure 2: Dataset of the model

II. Simulation

In modelling the queuing system for the train, different entities were created and they include the following:

a. Stations
b. Signaling Blocks
c. Trains

A. Stations

A Stations class was the first entity that was created to handle the starting and destination station of the train. It holds the (From and To) of the train information.

B. Signaling Blocks

A Blocks class is another entity that was created to handle creation of the signaling blocks for each journey. The create method in the Blocks class takes the argument station and creates the number of signaling blocks to the next station. It returns the number and stores it in the sigBlocks = [].

C. Trains

A Train class is an entity that contains information about the train and its processes is creates. The variables for the instance are declared using the constructor method (_init_), and the train is recorded from the dataframe for collecting simulation results. Based on the question and assumptions stated, it is assumed that the trains travel at the same speed which means the signaling blocks all have equal length. This is done in other to keep the travel time equal when testing the blocks so it does not affect the total travel time.

The train process is defined in the process method. First the train drives from the depot to the station London Old Oak Common. This is assumed to take 30seconds. Using the current location which is set as the variable here and the departure time is recorded in the simulation results dataframe created. For each of the destinations, the data is acquired from the dataframe, each train arrives from the depot, waits to depart, requests for the signaling block. If the signaling block is empty or not in current use, the train proceeds to enter. If not, the train waits till the signaling block is empty or turns green before it can proceed to use the signaling block. If the train is at the final station, the time of arrival for the journey is recorded.

III. Optimisation

The first part of the optimisation was to create a train schedule that sets out the departure and arrival time for trains with the first train leaving at 7 a.m. and the last train leaving

at 10 p.m. This was done in the Main function as shown in figure 3 below.

```
def main(n,tBlocks,start=7*3600):
    #timing depending on number of trains
    timing = round(3600/n,2)

    #start the simulation by 07:00:00
    yield env.timeout(start-env.now)

    #get all stations on the HS2 line
    stations = Stations.get_stations()

    for j in range(0,15):
        for i in range(n):
            t = Train(i,j,n, stations,tBlocks)
            env.process(t.process())
            yield env.timeout(timing)
```

Executed in 6ms, Started at 22:33:30

Figure 3: Setting Train schedule

The figure 4 shows the delay time that was introduced in the train schedule at 9 a.m. due to a temporary break down due to electrical malfunction. This was done using an if statement in the process function within the Train class. Since the time is set in seconds, the if statement used are seen in figure 4 and figure 5

```
    #Delay for 9am train
def process(self):
    if env.now >= 32400 and env.now <= 32431:
        delay = 1
    else: delay = 0
```

Figure 4: Train delay on 9am train

```
if delay == 1:
    print(f"{now():s} {self.name:s} THERE IS A 30MINS DELAY FOR THE 9AM DUE TO TEMPORARY BREAKDOWN")
    yield env.timeout(1800)
    delay = 0
```

Figure 5: Train delay on 9am train

The delay of the 9 a.m. train happens to affect the Train 0 as seen in the visual output below in figure 6. Train 0 could not request for block 1 at 09:04:32 a.m. until after 30 mins of wait.

```
09:00:00 [Train  0] drives from depot to starting station
09:00:30 [Train  0] Waiting to depart London Old Oak Common for Birmingham Interchange
09:00:30 [Train  0] requests for block 0 to Birmingham Interchange
09:00:30 [Train  0] enters block 0 for journey to Birmingham Interchange
09:00:41 [Train  6] requests for block 5 to Birmingham Interchange
09:00:41 [Train  6] enters block 5 for journey to Birmingham Interchange
09:01:54 [Train  8] requests for block 2 to Birmingham Interchange
09:01:54 [Train  8] enters block 2 for journey to Birmingham Interchange
09:02:04 [Train  5] arrives Birmingham Interchange
09:02:04 [Train  5] Train has reached its final destination
09:03:18 [Train  7] requests for block 4 to Birmingham Interchange
09:03:18 [Train  7] enters block 4 for journey to Birmingham Interchange
09:04:32 [Train  0] THERE IS A 30MINS DELAY FOR THE 9AM DUE TO TEMPORARY BREAKDOWN
09:04:43 [Train  6] requests for block 6 to Birmingham Interchange
09:04:43 [Train  6] enters block 6 for journey to Birmingham Interchange
09:05:56 [Train  8] requests for block 3 to Birmingham Interchange
09:05:56 [Train  8] enters block 3 for journey to Birmingham Interchange
09:06:40 [Train  1] drives from depot to starting station
09:07:10 [Train  1] Waiting to depart London Old Oak Common for Birmingham Interchange
```

Figure 6: 9am Train delay

Below output in figure 7 is seen as Train 0 requests for block 1 to Birmingham Interchange at 9:34:32 a.m. after 30 minutes of delay.

```
09:20:30 [Train  3] Waiting to depart London Old Oak Common for Birmingham Interchange
09:20:30 [Train  3] requests for block 0 to Birmingham Interchange
09:22:06 [Train  8] arrives Birmingham Interchange
09:22:06 [Train  8] Train has reached its final destination
09:26:40 [Train  4] drives from depot to starting station
09:27:10 [Train  4] Waiting to depart London Old Oak Common for Birmingham Interchange
09:27:10 [Train  4] requests for block 0 to Birmingham Interchange
09:33:20 [Train  5] drives from depot to starting station
09:33:50 [Train  5] Waiting to depart London Old Oak Common for Birmingham Interchange
09:33:50 [Train  5] requests for block 0 to Birmingham Interchange
09:34:32 [Train  0] requests for block 1 to Birmingham Interchange
09:34:32 [Train  0] enters block 1 for journey to Birmingham Interchange
09:34:32 [Train  1] enters block 0 for journey to Birmingham Interchange
09:38:34 [Train  1] requests for block 1 to Birmingham Interchange
09:38:34 [Train  2] enters block 0 for journey to Birmingham Interchange
09:38:35 [Train  0] requests for block 2 to Birmingham Interchange
09:38:35 [Train  0] enters block 2 for journey to Birmingham Interchange
09:38:35 [Train  1] enters block 1 for journey to Birmingham Interchange
09:40:00 [Train  6] drives from depot to starting station
```

Figure 7: 9am Train delay 2

Again, in order to introduce variability in the travel time since it is assumed that all the signaling blocks are of the same length, the Log-normal distribution was used as described by [1] but this was done with different variables such as $\mu = 2.55$ and $\sigma = 0.11$.

IV. Visual Output

The following contain all the visual output of the train modelled system.

The figure 8 below shows a visual output of the simulator which was used in testing the model. Using random values of 9 trains and 7 signaling blocks, we can see the output of the simulation. Train 0 starts from 7:00:00 a.m. and drives from depot to the starting station, there was a 30 seconds delay set which is seen at 7:00:30 a.m. waiting to depart London Old Oak Common to Birmingham Interchange.

```
In [16]: simulator(9,7) #no. of Trains, no. of Signalling blocks
```

Executed in 453ms, Started at 22:33:34

```
07:00:00 [Train  0] drives from depot to starting station
07:00:30 [Train  0] Waiting to depart London Old Oak Common for Birmingham Interchange
07:00:30 [Train  0] requests for block 0 to Birmingham Interchange
07:00:30 [Train  0] enters block 0 for journey to Birmingham Interchange
07:04:32 [Train  0] requests for block 1 to Birmingham Interchange
07:04:32 [Train  0] enters block 1 for journey to Birmingham Interchange
07:06:40 [Train  1] drives from depot to starting station
07:07:10 [Train  1] Waiting to depart London Old Oak Common for Birmingham Interchange
07:07:10 [Train  1] requests for block 0 to Birmingham Interchange
07:07:10 [Train  1] enters block 0 for journey to Birmingham Interchange
07:08:34 [Train  0] requests for block 2 to Birmingham Interchange
07:08:34 [Train  0] enters block 2 for journey to Birmingham Interchange
07:11:12 [Train  1] requests for block 1 to Birmingham Interchange
07:11:12 [Train  1] enters block 1 for journey to Birmingham Interchange
07:12:36 [Train  0] requests for block 3 to Birmingham Interchange
07:12:36 [Train  0] enters block 3 for journey to Birmingham Interchange
07:13:20 [Train  2] drives from depot to starting station
07:13:50 [Train  2] Waiting to depart London Old Oak Common for Birmingham Interchange
07:13:50 [Train  2] requests for block 0 to Birmingham Interchange
```

Figure 8: Random simulator

The figure 9 below shows the average delay time of the simulation calculated while using 9 trains and 7 signaling blocks. The result of the average delay time was 95.75 seconds.

Calculating Average Delay Time

```
delayTimes = setDelay(simulationData)
print(f"Average Delay Time was: {delayTimes}")
```

Executed in 36ms, Started at 11:51:18

```
Average Delay Time was: 95.75
```

Figure 9: Average Delay time

Due to the variations introduced, the figure 10 below shows the average travel time of the simulation calculated while

using 9 trains and 7 signaling blocks. The result of the average travel time was 1779.77 seconds.

Calculating Average Travel Time

```
travelTimes = getResults(simulationData)
print(f"Average Travel Time was: {travelTimes}")
```

Executed in 5ms, Started at 11:51:18

```
Average Travel Time was: 1779.77
```

Figure 10: Average Travel time

Figure 11 below shows that the smallest average delay times when keeping the signaling block constant at 4 and varying the trains from 1 to 14 per hour were found between 2 trains per hour to 5 trains per hour. Well, as the number of signaling blocks increased, the average delay time also increased. The train with the smallest average delay time per hour was found in 3 trains per hour with 81.02 seconds.

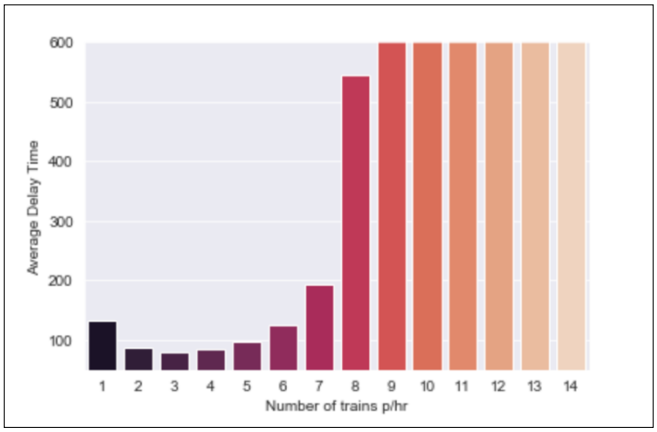

Figure 11: Average delay time for trains per/hour

In Figure 12 below, it shows the delay times for signaling blocks per hour. When varying the signaling blocks from 1 to 17 per hour and keeping the trains constant at 8, the smallest average delay times are between 13 signaling blocks per hour and 17 signaling blocks per hour. This implies that when the number of signaling blocks per hour increases, the average delay time decreases. The smallest average delay times for signaling blocks per hour is seen in signaling blocks 17 per hour with 59.16 seconds.

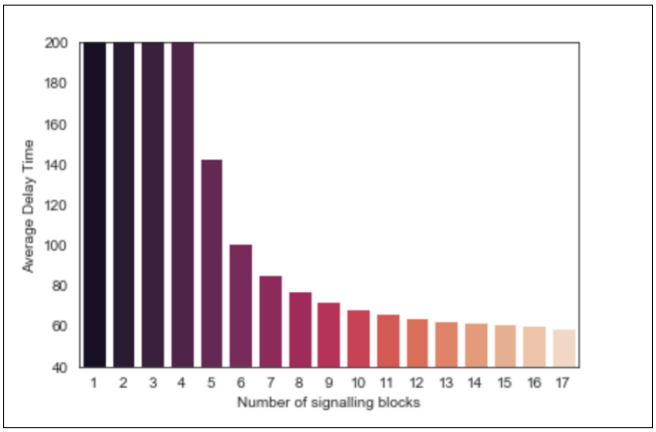

Figure 12: Average delay time for signaling blocks per/hour

When maximizing the number of trains operating per hour under the constraint that the average delay time should not be higher than half the scheduled time between consecutive trains. The Exhaustive search (brute-force), Monte Carlo and Greedy Hill Climbing techniques were implemented. The figure 13 below shows the result of the Exhaustive search algorithm. Using the minimum number of trains as 1 and the maximum number as 27 and a minimum number of signaling block as 1 and the maximum number 14 produced a maximized output at the 133rd execution with 13 trains and 9 signaling blocks per hour and a time of 12.72 seconds. This took about 4 minutes 55 seconds to execute.

	Train&Blocks	Time
133	(13, 9)	12.72

Figure 13: Exhaustive search output

While for the Monte Carlo algorithm for maximizing the number of trains per hour, using the minimum number of trains as 1 and the maximum number as 27 and a minimum number of signaling block as 1 and the maximum number 14 with 50 iterations produced a maximized output at the 20th execution with 13 trains and 14 signaling blocks per hour and a time of 12.88 seconds. This took about 47.8 seconds to execute. The output for the Monte Carlo is shown in figure 14 below.

```
MonteCarloOutputs
```

Executed in 3ms, Started at 12:09:04

```
(20, 11, 15, 12.88)
```

Figure 14: Monte Carlo output

The Greedy Hill climbing algorithm for maximizing the number of trains per hour, using the minimum number of trains as 1 and the maximum number as 27 and a minimum number of signaling block as 1 and the maximum number 14 with 50 iterations produced a maximized output at the 6th execution with 27 trains and 8 signaling blocks per hour and a time of 115928.93 seconds. Although, it was manually interrupted after 9.55 seconds. The output for the Greedy Hill climbing is shown in figure 15 below.

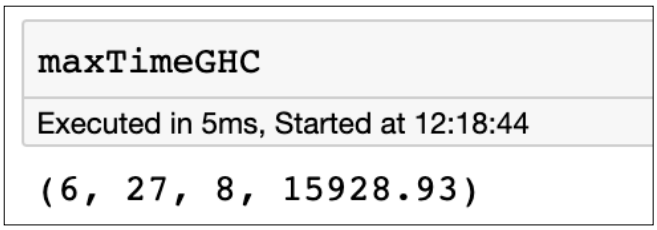

Figure 15: Greedy Hill Climbing output

When comparing the Exhaustive search and Monte Carlo algorithms together, the Monte Carlo executes faster than the exhaustive search with an execution 47.6 seconds. This is because the exhaustive search does every combination of trains and signaling blocks step by step before finding the maximised results for trains operating per hour. The Exhaustive search techniques gave the least maxisimed time of 12.72 seconds with 13 trains and 9 signaling blocks per hour. Followed by the Monte Carlo techniques which produced a maximised results for trains at 12.88 seconds with 13 trains and 14 signaling blocks per hour.

V. Conclusion

In conclusion, the modeling, simulation and optimization of the travelling speed of trains on the HS2 line from London Old Oak Common station to Birmingham Interchange station was achieved by inducing a temporary break down of 30 minutes due to electrical malfunction on the 9 a.m. train and introducing variability of traveling using a log-normal distribution.

References

[1] Y. Yang, P. Huang, Q. Peng, J. Li, and C. Wen, 'Statistical delay distribution analysis on high-speed railway trains', *J. Mod. Transp.*, vol. 27, no. 3, pp. 188–197, Sep. 2019, doi: 10.1007/s40534-019-0188-z.

[2] Stefan Scherfke and Ontje Lünsdorf https://simpy.readthedocs.io/en/latest/contents.html

[3] The SciPy community, "Optimize Linpro, Jul 23, 2020" https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html

[4] The Scipy community, "Log normal, Oct 18, 2015" https://docs.scipy.org/doc//numpy-1.9.1/reference/generated/numpy.random.lognormal.html

[5] C. Horn (Dr), "Simulating M/M/c Queue,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/658598/mod_resource/content/3/M-M-c%20v3%20-%20Jupyter%20Notebook.pdf (Week 8)

[6] C. Horn (Dr), "The MonteCarlo Problem,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/641931/mod_resource/content/0/MonteCarlo%20-%20Jupyter%20Notebook.pdf (Week 4)

[7] C. Horn (Dr), "Greedy Hill-Climbing,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/641947/mod_resource/content/0/GreedyHillClimbing%20v2%20-%20Jupyter%20Notebook.pdf (Week 4)

[8] C. Horn (Dr), "London Underground Step 1,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/669310/mod_resource/content/1/London%20Underground%20Step%201%20-%20Jupyter%20Notebook.pdf (Week 9 and 10)

[9] C. Horn (Dr), "London Underground Step 2 V4,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/669313/mod_resource/content/2/London%20Underground%20Step%202%20v4%20-%20Jupyter%20Notebook.pdf (Week 9 and 10)

[10] C. Horn (Dr), "London Underground Step 3 V4,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/669315/mod_resource/content/2/London%20Underground%20Step%203%20v4%20-%20Jupyter%20Notebook.pdf (Week 9 and 10)

[11] C. Horn (Dr), "London Underground Step 4,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/671587/mod_resource/content/1/London%20Underground%20Step%204%20v1%20-%20Jupyter%20Notebook.pdf (Week 9 and 10)

[12] C. Horn (Dr), "London Underground Step 6,2020" [online] Available:https://moodle.ncirl.ie/pluginfile.php/671595/mod_resource/content/1/London%20Underground%20Step%206%20v2%20-%20Jupyter%20Notebook.pdf (Week 9 and 10)