Network Model Generation

TU Delft
EPA1352 - Assignment 3
Group 6
Christiaan Ouwehand (4293053)
Ewout ter Hoeven (4493346)
Marlou Ceha (4691539)
Timo Frazer (4579992)
Zoé Huizing (4660455)

Introduction

The goal of this lab assignment is investigating critical road infrastructure in Bangladesh. This is done by calculating the delay for trucks driving on the N1, N2, and their N-class side roads longer than 25 km. Delays in the model occur due to potential bridge breakdown. The key focus of this analysis is the travel time of the trucks.
A useful tool for solving critical infrastructures is object oriented programming (Eusgeld & Nan, 2009).To analyze the travel time, an agent-based object oriented model, specifically the Mesa Python library (Kazil et al, 2020), will be used. The delays will be calculated for each truck starting or finishing on one of these N-roads. Every bridge has a different condition (from good to very bad) based on which it has a certain chance to break down in each of the 4 different scenarios. By letting trucks drive over the network and recording their travel times, the critical bridges can be identified to make a robust infrastructure maintenance policy.

Data preparation

There is a roads and bridge file provided as data for this assignment. In order to run the model for the analysis, these files were processed. Figure 1 shows how this has been done. An assumption for this assignment is that every road between two bridges is a straight line. This can be assumed so because the travel time gained from this is minor in comparison to the travel time lost by bridge delay.

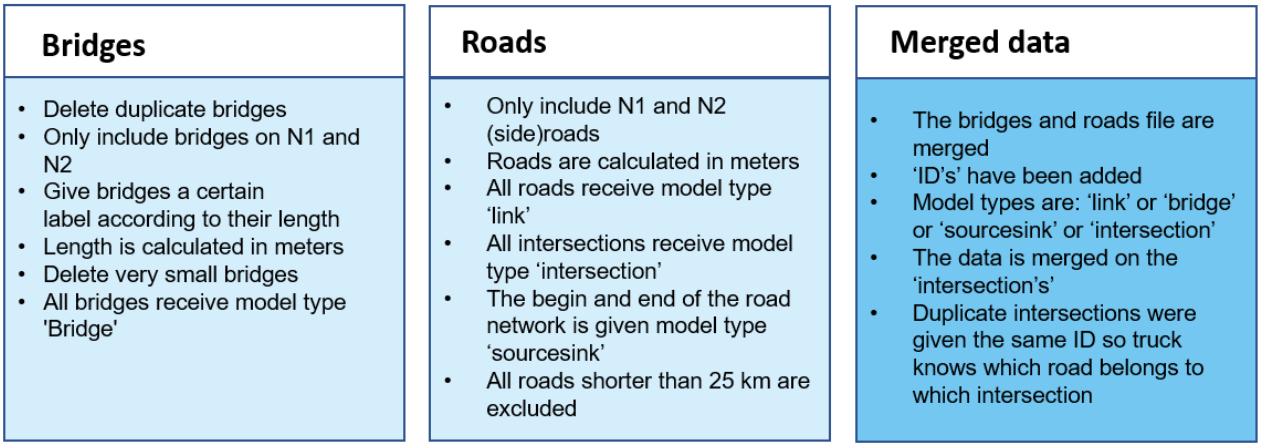

Figure 1: Data preparation

For this assignment, only the bridges and roads on the N1 and N2 were selected, with its N-class sideroads longer than 25 km. The roads and bridges were sorted to their longitude and latitude to give the right order. The lengths of the bridges and roads were calculated in meters and very small bridges were deleted. The variable model_type is added to assign the different infrastructural components (bridge, link, intersection and sourcesink). The intersection is assigned to the entry that gives the first row in the dataset of a specific sideroad. The 'sourcesink' is assigned to the last entry of the road. Every bridge was given a label according to their length. This label will be used later on in the simulation of the model to refer to a certain probability to break down. These probabilities will vary per scenario.

Methodology

As in lab assignment 2, the trucks are generated from a source to a sink every 5 minutes using the Python Mesa package. While driving, these trucks can encounter a broken bridge. An overview of the model implementation of the breakdown of each bridge categorie and the scenarios can be found in Appendix A.
To add all the roads longer than 25 crossing the N1 and the N2 a graph of the network is made. For that, a NetworkX (Hagberg & Conwaymodel, 2020) of the dataset is generated at the model initialisation in the BangladeshModel. To make the graph each infrastructure object ID is added as a node. Every node is undirected linked with its neighbor. These edges of the graph are weighted with the length of the infrastructure object they are representing. The code to create the graph is shown in Appendix B. All roads that fit the condition and will be analyzed can be seen in Figure 2.

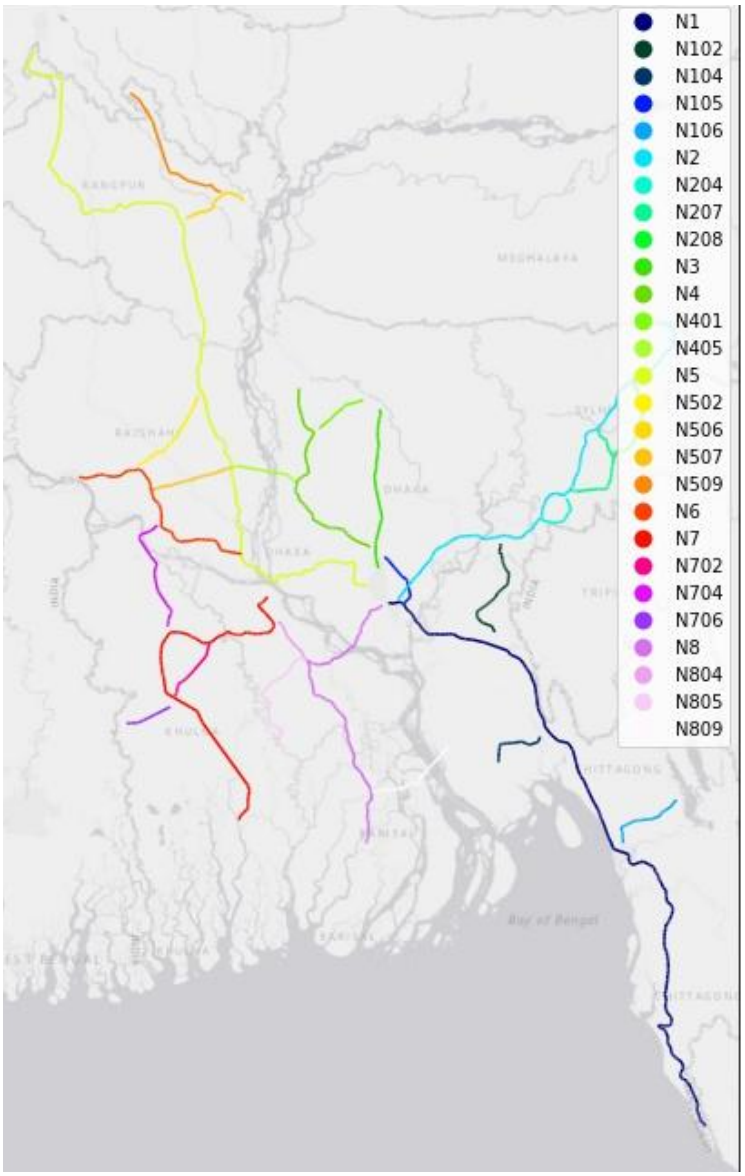

Figure 2: Map of the selected roads in Bangladesh

To make the truck travel from a random source to a random sink on the graph the trucks make use of Dijkstra shortest path algorithm (Dijkstra, 1959). The code used for the shortest path is added to the get_straight_route function and can be found in Appendix B. The code uses the Source and Sink ID to return a list of ID (from roads and bridges) the truck has to travel on. In the data, the intersections between two roads have been given the same ID. Once a truck is driving on an intersection, it will take the following shortest road.

To generate results, every scenario (Table A2) is replicated ten times. With these replications, the average and the 95% confidence interval of the delay, the density of the delays and the outliers will be analyzed.

This model has a discrete simulation component. In the Mesa model this leads to recalculation at every time step owing to a model run time of a few hours. To fasten the run time, every print statement in the model is commented.

Results

To analyze the differences between every scenario, the recorded average travel times were analyzed. The 95% confidence interval bounds and range were calculated and histograms and boxplots were made for every scenario. These plots can be found in Appendix C. The average travel time and the 95% confidence interval per scenario can be found in Table 1.

Table 1: average travel time and 95% confidence interval of the trucks in each scenario

Secenario	Average (min)	95% confidence interval			Interval range in function of the average travel time
		Low bound (min)	High bound (min)	Interval range (min)	
1	230.7	229.6	231.9	2.4	1.00%
2	302.2	300.7	303.8	3.0	0.99%
3	865.5	861.4	869.6	8.2	0.94%
4	1256.7	1251.2	1262.2	11.0	0.87%

From Table 1 can be seen that the delays are increasing by the scenario's. This is an expected result because the breakdown probability of bridges is also increasing over the scenarios.
Notably, under ten replications, the confidence interval is narrowing if the delay time is increasing. This is also expected, since as the chance increases for bridges to break down, more will break down, which results in lower variations between replications (law of large numbers).
For scenario 1 and 2, the variation in average travel time is around 1% of the average modeled travel time. For scenario 3 and 4 this is lowering up to 0.87% meaning the travel time is more predictable.

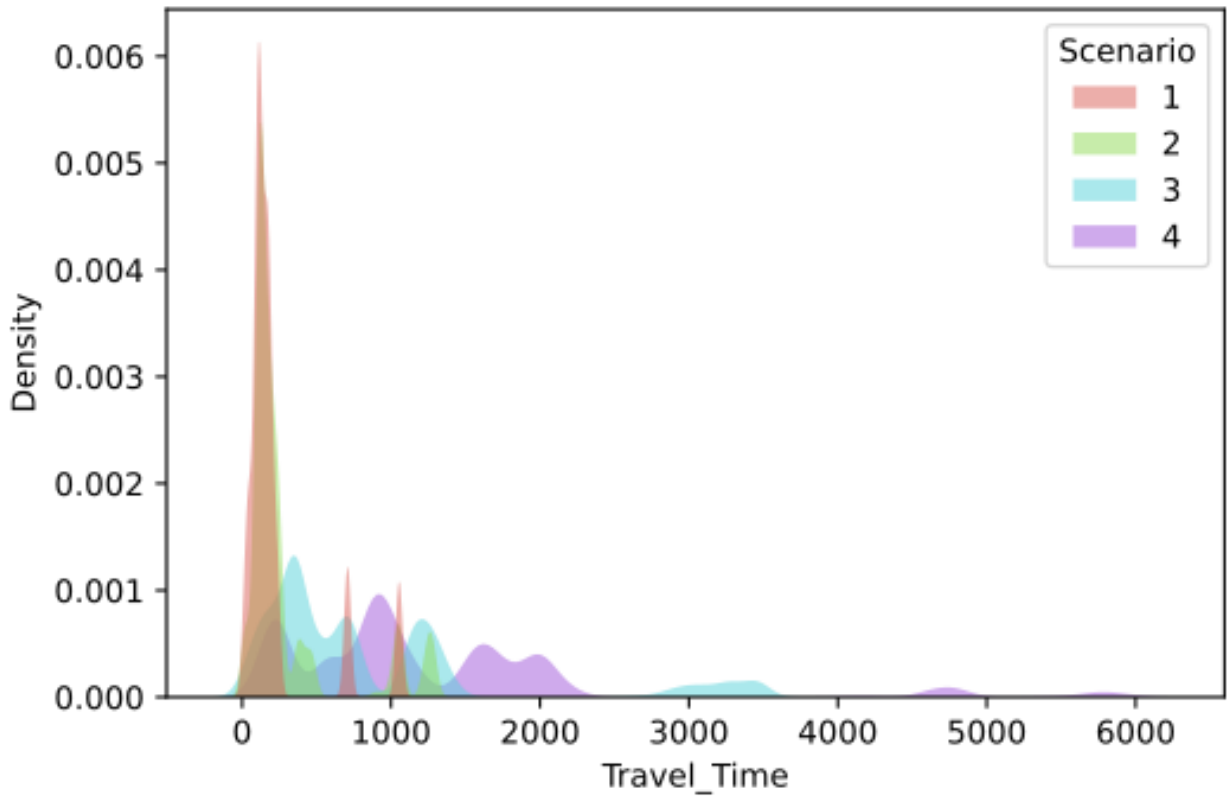

Figure 3: Kernel Density Estimation (KDE) of the travel time for the scenarios

From the boxplots from Figure C2 can be seen that if the scenarios are increasing the amount of outliers is also increasing. These outliers can have an impact in the average travel time from Table 2.
From Figure 3 and C1 can be seen how the travel time density is divided. For the first two scenarios the travel time is approaching a normal shaped distribution. For the last two scenarios the travel time is more uniformly divided. This makes it more difficult to make a good prediction of the travel time. For scenario 1 the delay time range is around 1000 minutes, this increased to 1200 minutes for scenario 2, 3500 for scenario 3, and 6000 for scenario 4.
This can be explained by scenario Table A2; the probability of breakage of bridges of category C and D is doubling when scenarios are increasing. Hence, scenario 4 is the only scenario where bridges of categorie A could break down. Classified A bridges are the most common in the model.

Conclusion and limitation discussion

The probability of bridges breaking down will under different scenarios generate delays for trucks on the N1, the N2 and their side roads longer than 25 km in Bangladesh. In this research it is shown that if the travel time increases the 95% confidence interval becomes narrower by 13%. This narrowing effect could be even larger if taken into account that for scenarios where there is a large probability of bridge breakdown (scenario 3 and 4) the differences in travel time can be up to six times larger than normal.

This study has also a few limitations. First, all the bridges break at the start of the model at the same time. Once the model is set up, no more bridges are breaking down.
This means that a truck that is later generated in the model has a lower probability to be stuck by a broken bridge than a truck generated earlier in the model. This is because a bridge is repaired after an amount of time steps. Second, the shortest route a truck is currently taking is based on the shortest travel distance and not the shortest travel time. This does not give trucks the possibility to take a detour that could save some travel time. Third, no analysis has been done with suppressed outliers to evaluate their impact on the average travel time and the confidence interval.

In a follow-up model these limitations could be taken into consideration. For further research it could be modeled that the chance of bridge breaking down is calculated every time step. Also, when a bridge is fixed, the condition of the repaired bridge needs to be in the A category as it now stays in the same category.

References

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische mathematik, 1(1), 269-271.

Eusgeld, I., & Nan, C. (2009, December). Creating a simulation environment for critical infrastructure interdependencies study. In *2009 IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 2104-2108). IEEE.

Hagberg, A., & Conway, D. (2020). NetworkX: Network Analysis with Python.

Kazil, J., Masad, D., & Crooks, A. (2020). Utilizing Python for Agent-Based Modeling: The Mesa Framework. Social, Cultural, and Behavioral Modeling, 308–317.

Appendix A

To implement the delay time of a truck in the different bridge classes a definition *get_delay_value* is created. The diverse types of bridges are categorized as follows:

Table A1: delay per bridge from each category

Bridge Length	Category Name	Delay time for a truck
Over 200 m	XL	Triangular (1,2,4) hours
Between 50 and 200 m	L	Uniform (45,90) minutes
Between 10 and 50 m	M	Uniform (15,60) minutes
Under 10 m	S	Uniform (10,20) minutes

To get the delay time all in the same unit, the delay time of category XL was converted from hours to minutes, resulting in Triangular (60,120,240) minutes. A dictionary was used to obtain the delay times for the different bridge categories.
In the model, every bridge also has distinct categories of conditions for the bridges. If a bridge is in a good condition, it will be categorized as A. The bridges in the worst condition will be categorized as D.

Table A2: Scenarios and percentage for each bridge to break down.

Scenario	Cat A %	Cat B %	Cat C %	Cat D %
0	0	0	0	0
1	0	0	0	5
2	0	0	5	10
3	0	5	10	20
4	5	10	20	40

Scenario 0 is in this model the base case to compare the delay generated by the breakdown of bridges.

Appendix B

```
def create_network(self, df):
    graph = nx.Graph()
    for row_index in range(0, len(df)):
        id1 = df.loc[row_index]['id']
        id2 = df.loc[row_index]['id'] + 1
        graph.add_edge(id1, id2, length=df.iloc[row_index]['length'])

    for row in df.index:
        graph.add_node(df['id'][row])

    return graph
```

Figure B1: Code to generate the road graph

```python
def create_path(self, source, sink):
    path = nx.shortest_path(self.network_graph, source, sink)

    self.path_ids_dict[source, sink] = path
```

Figure B2: Code for shortest path

This path is saved into path_ids_dict. In the set_path() method in Vehicles, the path_ids_dict is checked if the values do exist or not or not, if so, then the existing path is returned, if not, then the get_route() method is asked.

Appendix C

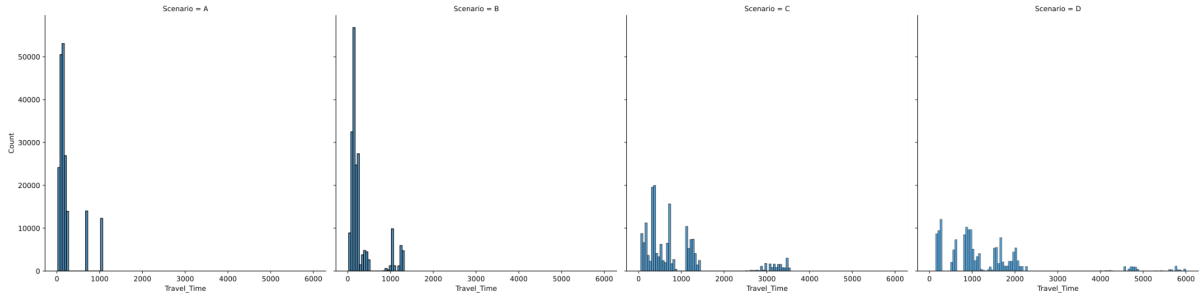

Figure C1: Histograms of every scenario

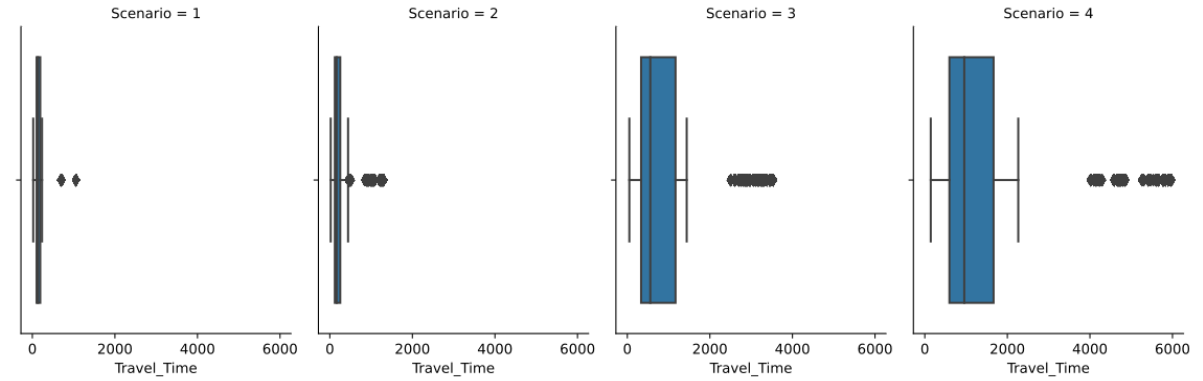

Figure C2: boxplot of every scenario