Data quality of the Bangladesh infrastructure database

Thomas Sandbergen 4720814
Thomas Rous 4963946
Matthijs van de Wiel 4896947
Yassine Mouhdad 5879566
Tom Hillenaar 4861973

March 24, 2023

1 Introduction

The goal of the task given by the World Bank is to find out critical and vulnerable parts of the infrastructure of Bangladesh that should be invested in. Natural disasters can cause critical infrastructure to get damaged and break which has a great impact on the country's logistics and the general economy. The road infrastructure in Bangladesh is a massive connected network of roads, bridges, and culverts which asks for a complex model. In order to accurately analyze this system, vehicles are simulated driving the roads to see how much delay time is formed. There are different possibilities for modeling such complex problems with interaction between agent sets. One of them is agent-based modeling (ABM). The focus in ABM is on individual agents (Klabunde, 2016). This includes their interaction with other agents, the effects of that interaction, and how this will affect model outcomes. The different interdependencies in transportation (roads, bridges, and vehicles) are complex and ABM allows for micro-data to be micro-simulated (Macal & North, 2005). Since the infrastructure concerns many different instances interacting on a micro-scale which together make up the whole infrastructure network, ABM should be appropriate in this case. The 'Mesa' module is used in Python for ABM modeling.

Within this research, the project scope is the N1, N2, and their respective side roads larger than 25 kilometers. Connecting these roads creates an infrastructure network that includes trucks driving to different destinations in both directions. The NetworkX module is used to create a network that can be analyzed and help vehicles find the shortest path to their destination. The parts calculated by the NetworkX module are used in the model created with the 'Mesa' module. In order to get the most useful representation of the road infrastructure network, the available data has been cleaned and prepared. However, the data is not yet fit for the simulation model as it is. In Chapter 2, the data preparation procedure is explained to fit the data according to the requirements. Chapter 4 gives elaboration on the model structure needed for this assignment on NetworkX and every individual component. Chapter **??** explains the outcomes of vehicle drive times under different scenarios. Chapter **??** contains the discussion within the report and includes an explanation of assumptions made, potential improvements, and limitations of the model.

2 Data Preparation

Before the model is used, data preparation was necessary on order to collect and reshape data in the correct format for the model. In order to give a structured overview of the data preparation a table is given with a row for each part of the data preparation process.

Table 1: Overview of cleaning within data preparation process

Subject	**Method of data cleaning**
Understanding of essential information for roads and bridges in the simulation model	Eight columns are selected to construct datasets useful for simulation, containing content of the road, id, model type, condition, name, lat, lon, gap, and length.
Selecting datasets	The file 'roads3.csv' contains most of the information necessary for the final dataset. However, bridge-type information is missing from this dataset. Information on bridges is retrieved from 'BMMSoverview.xlsx'.
Create correct length for roads.	Take the differences in chainage values and multiply this number by 1000 in order to get the length in meters.
Selecting side of bridge	Each bridge contains two points: Bridge start (BS) and bridge end (BE). We selected only BS and, therefore, dropped BE.
Dropping column 'gap'	The column 'gap' is not necessary anymore and, therefore, is dropped out of the dataframe.
Renaming column	In order to merge the datasets later on, the column lrp in the roads dataframe has been renamed into 'LRPName'.
Drop duplicates	Duplicate bridges are dropped out of the dataframe. The column used is 'LRPName'.

Merging the two datasets and selecting the necessary columns again	The dataset is mainly based on the 'roads3' file since this file is the most extensive file of the two sets selected. The idea is to add specific information on bridges. For merging the two datasets, the columns 'road' and 'LRPName' are indicators. Some columns are duplicated in the merged dataframe, since both dataframes contain these columns. Therefore, some duplicate columns from origine the BMMSoverview file are being dropped. Important columns added from 'BMMSoverview' are: damage conditions and length of bridges. The column length, however, is also present in the roads file. These columns are, therefore, combined into one column since information is necessary from both columns.
Selecting roads	Selecting the roads within the 'road' column. Each road which starts with either N1 of N2 will be selected. After this process. Every road shorter than 25 km is dropped out of the dataframe.
Completing conditions of bridges	Every bridge should contain a condition (either A, B, C or D). However, some bridge do not contain a condition. These bridges will have assigned conditions. In order to do this, the distribution between bridge conditions has been used. Each bridge with missing condition will have a probability to have either condtion A, B, C or D assigned.

Implementing ID number	The dataframe does not contain an id number for each part of the network. This is necessary for the model to run and, therefore, has been added to the model. The first datapoint has id 0, the second id 1 etc.
Renaming type of bridges	There are currently either 'Culvert' or 'Bridge' type of bridges in the dataframe. Both type of bridges are implemented in the model and will have the 'Bridge' type.
Applying the setup of the demo file	The column structure of the demo file has been applied on the dataframe in order to make sure that the model can run with NetworkX
Renaming specific datapoints into either sourcesinks, intersection or link	An important asset in the third assignment is the column: 'type'. However, after scanning the data set, it could be seen that several changes had to be made. For instance, if 'CrossRoad' was indicated it must be changed to 'Intersection'. Another example is the change from 'Culvert' and 'Bridges' to just 'bridges'. In this way, 'type' only consists of the following four values: 'sourcesink', 'link', 'bridge', and 'intersection'. An important addition to this assignment was that a starting spot of a truck would also be an ending spot and the other way around. Therefore, any column with source and sink type has been changed to just 'sourcesink' since it is the same.
Adding links in between two bridges	Sometimes there are two bridges nexst to each other in the dataframe. A link with length 0 has been added in between these bridges.

Replacing some specific intersection points	Some intersection points were located on the wrong location and, therefore, have been replaced manually.
Create overlap between intersections	In order to let intersections connect with each other, their ID must be identical. The system used for this phenomenon is based on the first intersection point of the two roads. A crossroad exists out of two roads with each an intersection point in the dataframe. The id of the first intersection point in the dataframe of a crossroad is the base for the id of the second intersection point of the crossroad. In order to
Combining links into one large link	Sometimes multiple links are located next to each other in the dataframe. These links are combined into one larger link to maintain the structure of link - bridge - link - bridge etc. (obviously, sometimes there is an instersection point in between.)

3 Model Structure

The model creates an accurate representation of the N1, N2, and their respective side roads with infrastructure components where vehicles travel across. First, the general model structure is shortly explained as it consists of different files interacting with each other and responsible for different processes and producing in- and output. Second, the class structure is elaborated to give a better understanding of the agents in the model. The last section highlights the vehicles and how they operate while calculating and traveling the network.

3.1 File structure

There are six files in the model structure responsible for different tasks. How these processes are linked and what every file produces is shown in figure 3.1 on the next page. The difference between model-run.py and model-viz.py requires elaboration since the process is similar but has important differences. The files both run the model with a different setup. Model-viz visualizes the Mesa model with a single manually set scenario of bridge conditions, which will be further explained in the next section 3.2. Model-viz requires a manual start and stop in the external web browser visualization. Model-run is able to run multiple scenarios and of each scenario multiple replications once the file is run and stops when the number of scenarios and replications are reached. The input for the files is the same produced by the files model.py, components.py, and create-input-data-roads.py. The process for create-input-data-roads.py was explained in section 2. The components.py and model.py file will be elaborated in the next section 3.2.

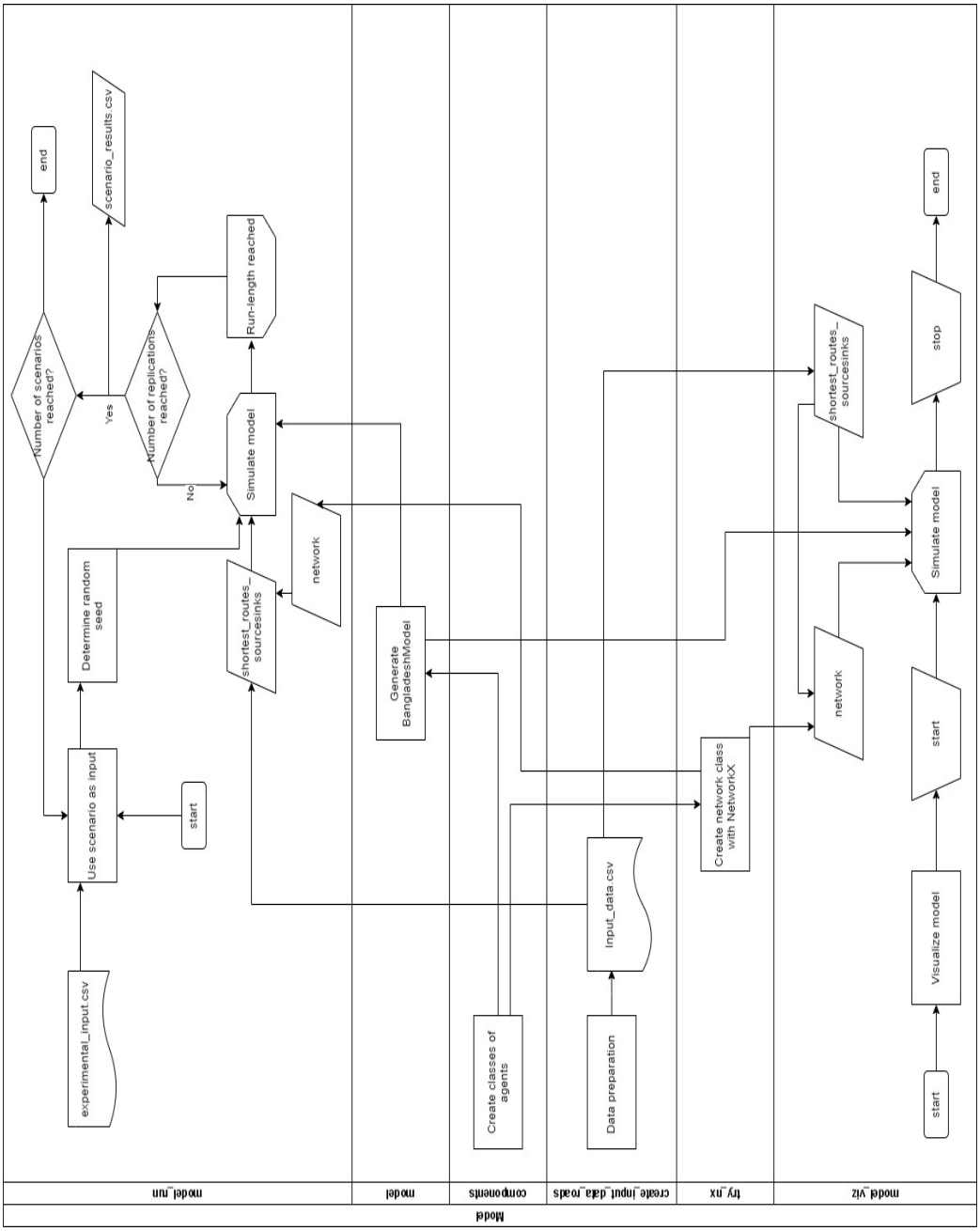

Figure 1: Swimlane diagram of the different files in the model

3.2 Class structure

To create the different agents in the model produced by model.py, components.py builds different classes of infrastructure agents and vehicle agents. Figure 2 gives a clear overview of inheritance and how the classes are related to each other. The Model and Agents classes are used from the Mesa module. These classes show the necessary input and functions to build a Mesa model which does not add additional information to the figure. Their specific characteristics, functions, and values are left out of the diagram since they are "inherited" as a normal class would but they are specified in a more specific way in the BangladeshModel class and Infra and Vehicle classes. The classes from the Mesa module show the necessary input and functions to build a Mesa model which does not add additional information to the figure.

Moreover, the SourceSink class inherits methods and properties from both the Source and Sink class. This means that these agents are able to create and remove vehicle. This is one of the important elements that allows the model to create and remove vehicles that can travel in both ways.

The most complex component in the model is the Bridge class. The Bridge class is what creates delay time in the model based on the condition. Links or roads do not create delays. This condition (A, B, C or D) is gathered from the data frame per bridge. The idea is that condition A is the best state of a bridge and condition D is the worst. The bridges have a chance of breaking down according to this condition. The vehicles are affected by this delay time and have to wait accordingly at a bridge before they can travel across.

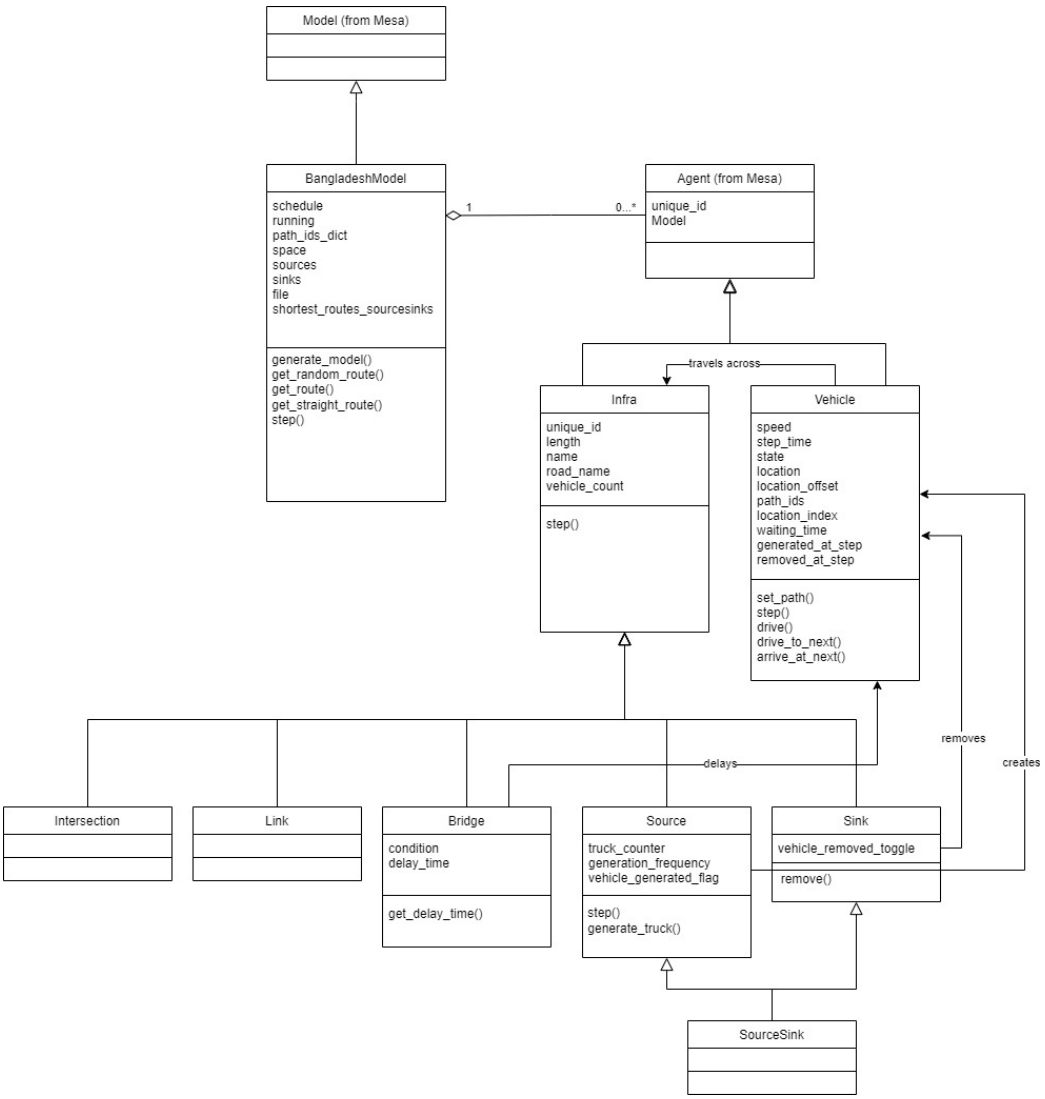

Figure 2: Class Diagram

As seen in figure 2, the vehicle class has associations and relations with multiple infrastructure agents. These are elaborated further in the next section 3.3.

3.3 Vehicles

As seen in the in the previous section 2, vehicles are agents in the model that interact and have relations with the infrastructure agents. Vehicles are created by Sources, removed by Sinks and are delayed by Bridges. They travel across the network via the different infrastructure agents. However, the input to travel the network and find the shortest path from the SourceSink that created to the vehicle to the destination SourceSink is retrieved from another model created with the NetworkX module. Which processes and input are created by the Mesa module and NetworkX module is shown in figure X.

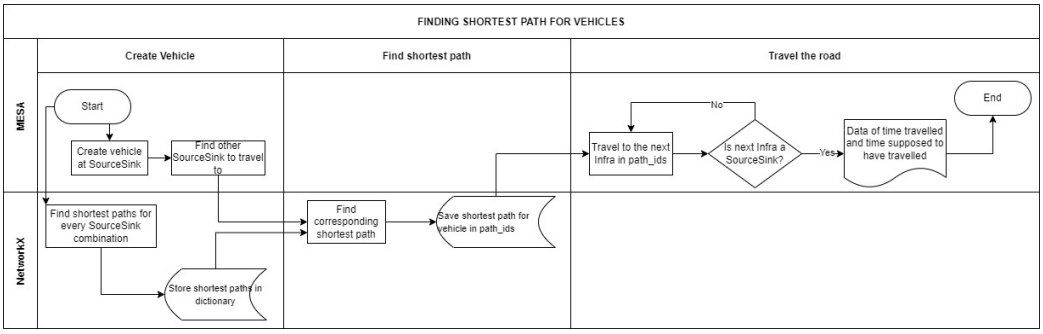

Figure 3: Flowchart of the relation between Mesa and NetworkX in the Vehicle process

How the NetworkX model is created, will be elaborated further in the next section 4.

4 NetworkX Explanation

As explained in the introduction, NetworkX has been used to calculate the shortest paths between different beginning and end points (SourceSinks). The method is to create a graph network of the N1, N2, and the sideroads connected to these roads. This section elaborates on this part of the model.

First, the input data of the road network is imported. This data has to be transformed into a graph with nodes and edges. However, the data is not yet fit for constructing a network as described. In the data, all the LRPs are points that have a length, which could make them both a node and an edge. The assumption is made that every point in the data becomes a node. An edge is drawn between two nodes in the network for the length of that point. This means that the two nodes and an edge display one point in the original data. For the last edge in the network, the length of the last two nodes is combined, as the last node would not have a length otherwise and produces errors. Looping through all the points and connecting the nodes with edges gives a network of the N1, N2, and side roads. All nodes receive the name of their id number. This allows for intersections to be created properly. ID numbers overlap between for instance the N1 and the N2. If the id already exists, the loop draws an edge toward that node instead of creating a new node with the same id. This connects all the roads together.

The next step is to create a set of all the SourceSinks based on the model_type from the original data. With the shortest path function from NetworkX, the shortest paths are determined for every SourceSink combination and stored in one large dictionary. This dictionary contains the paths in both directions of the combination of SourceSinks, which can differ due to parallel bridges that go in different directions and have a different delay time. The dictionary is assigned to the Bangladesh Model for every new replication. Within the model, the vehicles access this dictionary to find the combination of their starting SourceSink and destination SourceSink. The path found from this dictionary is how they travel within the Mesa model.

5 Results

The results chapter is divided into two subsections: the first subsection is information on the bridge class which is the input of the different scenarios. In addition, the different scenarios are listed. The second subsection includes the actual results of the model tested under different scenarios. Statistical data and visualizations have been made to interpret the results.

5.1 Bridges

As explained in section 3.2, the bridges have a chance of breaking down according to a condition. To analyze the network and show the impact of bridges breaking down, different scenarios are introduced with replications to test the model. The percentages that give the chance of a bridge breaking down per category of condition are displayed in Table 2 going from best to worst. This creates a different version of the network for every replication of a scenario, allowing for multiple replications to run and comparisons between them.

Table 2: Chance of bridges breaking down per scenario

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

Every bridge is either 'broken' or not after the scenarios are run. The delay time is assigned to every bridge according to the length of the bridge. For every range in length, a different stochastic distribution is used to determine the number of minutes every bridge delays the vehicles to cross. The distributions are shown in Table 2.

Table 3: Delay time per type of bridge

Bridge length	Delay time for a truck
under 10 meter	Uniform(10, 20) minutes
Between 10 and 50 m	Uniform(15, 60) minutes
Between 50 and 200 m	Uniform(45, 90) minutes
Over 200 m	Triangular(1, 2, 4) hours

The delay time will eventually function as waiting time for the vehicles. Bridges breaking down according to different scenarios and running the model based on these scenarios is described in the next section.

5.2 Scenario analysis

The goal of the assignment is to test the different drive times of trucks on multiple routes with different destinations under different scenarios. Different scenarios are important to test with the model since Bangladesh's climate disasters may have relatively small or enormous consequences for its infrastructure. Therefore, the model is tested for four scenarios including a 'base' scenario which includes no bridges breaking down. In order to assess the results, box plots have been made for visualization. Box plots are useful to see the spread of data points (Insightoriel (2022)). Finally, before interpreting the scenarios it must be stated that the model uses stochastic functions in order to calculate the number of broken bridges. Therefore, the scenarios differ for each run of the model. In order to be able to investigate the distribution of the output, more replications must be applied for each scenario. In this research there are 10 replications applied.

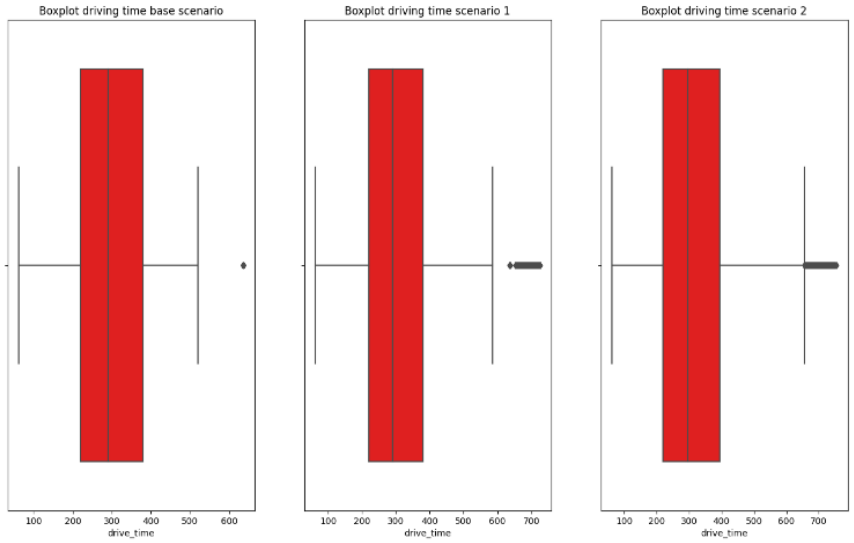

Figure 4: Boxplots for scenario 0, 1 and 2

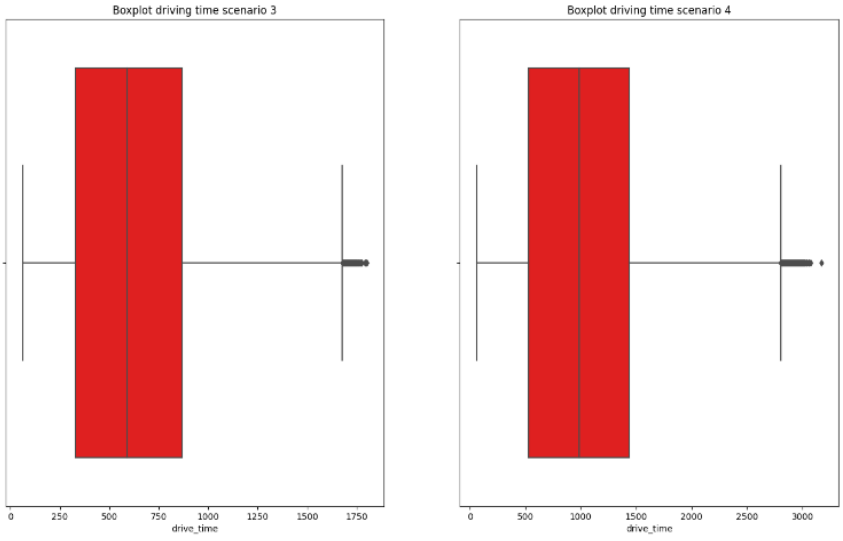

Figure 5: Boxplots for scenario 3 and 4

In order to make accurate interpretations different statistical data has been calculated for the drive time under different scenarios. Table 4 shows the mean, standard deviation, minimum value, different quartile values, and maximum value.

Table 4: Statistic results of the different scenarios

Scenario	base	1	2	3	4
count	96507	96474	96360	91952	85571
mean	305	308	315	621	1024
std	139	142	149	360	624
min	62	62	62	62	64
25	220	220	220	328	528
50	290	290	295	590	988
75	380	380	394	868	1439
max	637	726	755	1802	3169

The boxplots represent the behaviour that can be expected from the design of the experiments: the boxplots widen as the probability of bridge failure increases. In scenario 1, relatively few bridges still break down and this ensures that all times with broken bridges provide outliers. In the other scenarios, there are still outliers for the largest delays, but the number of outliers does decrease relatively. It is noticeable that minimum journey times remain low in all scenarios. This may be due to a very short route with very few bridges. If there are few bridges then with high probability bridges may still not break down. For the maximum time it takes trucks, you can see very clearly that this increases as the probability of bridges breaking increases.

All in all, it can be concluded that drive times on the roads are much dependent on the number of bridges being vulnerable to causing delays. Furthermore, the maintenance time for repairing bridges has a significant impact on delay times. As expected higher chances for the same type of bridges being unavailable cause higher delay times. However, it seems that an increase in the number of bridges being vulnerable to breaking down has more impact than the chances increase in the same number of bridges. Having more points on the road vulnerable to breaking down weighs heavier than existing vulnerable points becoming more vulnerable. However, more research must be done to verify this statement.

5.3 Bonus section

To obtain more accurate locations for the intersections, Shapefiles has been used. The latitude and longitude data in the model could namely be inaccurate at some intersections. Using this method, Figure 6 is produced that shows all the roads in Bangladesh according to the Shapefile and the intersections of relevant road sections (in red).

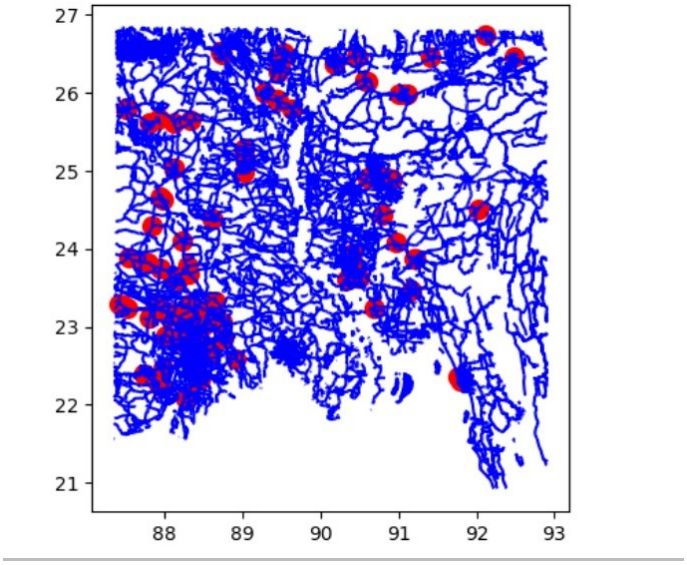

Figure 6: Locations of relevant intersections with Shapefiles

Using this overview, it can be checked whether the latitude and longitude of intersections in the original model are correct. For example, the intersection between the N1 and N2 have latitude 23.7060833 and longitude 90.5215271, according to our model. (This, and other coordinates, can be verified in the files provided with this report.) According to Figure 6 (the Shapefiles), there is indeed a relevant intersection (red dot) at that point, if one follows the axes. Comparing this in a visualization is, of course, not very precise. In further research, this validation needs to be done more extensively and in depth, by comparing the coordinates of every intersection.

In another approach to validate the locations of the model intersections, intersections in the model are compared to the actual location of intersections according to Google Maps. The latitude and longitude were inserted to

produce the red pointer below in the figures. Some of the results of this validation process are shown in figures 7, 8, 9, and 10 below.

Figure 7: Intersection between N1 and N2

Figure 8: Intersection N1 and N104

Figure 9: Intersection N2 and N207

Figure 10: Intersection on N2

It can be seen that for the intersections displayed here, the model intersections are spot-on. Also, this can be examined more extensively in further

research. A part of that is already done in Appendix A, section 7.1.

According to these two small validations studies, the conclusion is that the model outcomes will not change if the 'real' intersection locations are used, as the model intersections seem to be correct.

6 Discussion

In stochastic simulation experiments, at least thirty replications per scenario are the minimum, indicated by the central limit theorem, stating that this number is the least required to ensure normally distributed results (Frost, 2022). Due to time limitations, this assignment only managed an insufficient ten replications. This means that outliers (points in the tail of the distribution) occur more when displaying the results through a boxplot. Therefore, doing ten replications is not reliable enough. The boxplots in Figure 4 and 5 show this as well. For example, scenario 2, does have many outliers. With only ten replications, distinguishing this from coincidence or a pattern is not possible. In conclusion, the median and other quantiles can differ heavily, when running more replications. In total, the model has been running three times with different seeds to experiment whether the results differ much. The results did actually differ quite substantially. However, the overall picture of the effect of scenarios on driving time did stay the same. It is interesting for further analysis to run an appropriate amount of replications and see if and how the results differ from the results in this report.

A potential limitation of the model is in running time for the experimental setup. Namely, as a follow-up to the above, it would be logical to run more replications and scenarios. The running time for one run/replication with only one scenario is 1.5 minutes. The current experimental setup (with only 5 scenarios and 10 replications per scenario) has a running time of about 1 hour. In order to improve the results, as mentioned above, the number of scenarios and replications must be increased which will increase the total running time. It would be possible to execute the above additions and still have a reasonable running time. However, the model only contains the N1, N2, and a few side roads. When the complete road network of Bangladesh is implemented, the running time will probably be be too long. It is important to reflect on this. This means that other experimental design methods should be used to obtain accurate results. A concept that can be used is space-filling designs (Sanchez & Wan, 2021). Instead of running every possible scenario (full-factorial design) or only a few constructed manually, a systematic selection can be used. Currently in the model, the scenarios assign bridges different 'caused-delay' parameters, based on five different scenarios. For accurate scenario analysis, many more scenarios with different parameters must be tested. Sanchez and Wan (2021) describe this as a pitfall, when a handful

of scenarios is chosen which may not address fundamental questions. Space-filling designs could again be useful. It is important to consider this in further research.

Another limitation is found regarding the merging of the files. In the data generation process for the model, the road and BMMS (bridge) files were merged. The method used for this is 'right-merge', which resulted in the road file being the 'main file' and based on overlapping road and LRP, the records from BMMS were merged with those of the road file. The road file fitted the required data format better and was more complete. Moreover, the roads file had fewer double identifiers (LRPs) for different objects in comparison with the BMMS file. This way of merging makes sense since the structure is built up by links (road parts) and bridges in the roads file. More information was needed from the BMMS file on the bridges breaking down which was added to the roads file. Intuitively, this is also the correct way, because bridges are on (or nearby) roads. If we merged the other way around (left-merge), some parts of the road would not be taken into account. The road would simply stop at some point or not be connected, which is not an option. An inner merge would not work in this case as the road parts in the road file would not be taken into account. This is because only the bridges in the road file are merged with those in the BMMS if the LRPs are consistently used in both files. This leaves only bridges in the data frame. An outer merge is not chosen, occasionally the same bridge name does not have the same LRP in both files. With an outer merge, both these bridges (the same bridges, but with different LRP) will be included in the data frame used for the model, which is incorrect.

Therefore, we used a right-merge. However, we did not specify that only the left or right bridges should be taken into account from the BMMS file if the LRPs are exactly the same and overlap with a record in the roads file. Instead, the first record in the BMMS file with the overlapping LRP for the bridge is merged into the data frame and used in the model. In general, this fits the assumption of vehicles driving two ways better because sometimes the right bridge is pulled from the data frame and sometimes the left bridge is. If trucks drive in both directions, this is the way to go. This is the case since starting and ending nodes are one and the same in this model. However, separate nodes could be made for left and right bridges since in reality when for instance, the left bridge breaks down, the right lane traffic can continue driving. This would be an improvement that does require separate nodes for bridges left and right and a distinction for trucks that the direction implies

whether it can use a certain node. This change would greatly complicate the model.

To accurately conclude the impact of broken bridges on the road network in Bangladesh, more research is necessary. However, this report provides an insightful starting point.

References

Frost, J. (2022, 2). *Central Limit Theorem Explained.* . (Online; accessed 10 March 2023)

Insightoriel. (2022). *What Is Boxplot — Box And Whisker Plot — 5 Advantages Of Boxplot — Create Boxplot In Excel.* . (Online; accessed 10 March 2023)

Klabunde, W. F., A. (2016). Decision-making in agent-based models of migration: State of the art and challenges. *Eur J Population*. doi: https://doi.org/10.1007/s10680-015-9362-0

Macal, C., & North, M. (2005). Tutorial on agent-based modeling and simulation. In *Proceedings of the winter simulation conference, 2005.* (p. 14 pp.-). doi: 10.1109/WSC.2005.1574234

Sanchez, S. M., & Wan, H. (2021). Work smarter, not harder: A tutorial on designing and conducting simulation experiments.. doi: 10.1109/WSC.2015.7408296

7 Appendix

7.1 Appendix A

This appendix shows more screenshots of results of the validation process as described 5.3.

Figure 11:

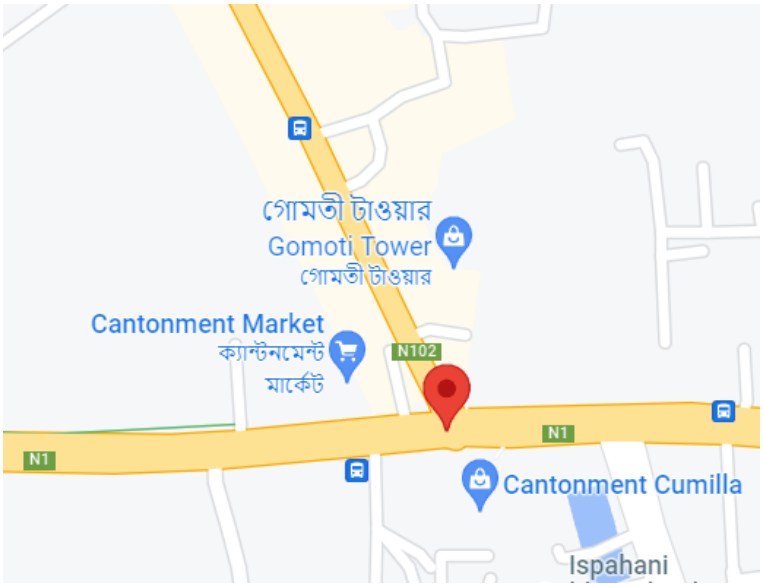

Figure 12:

Figure 13:

Figure 14:

Figure 15:

Figure 16:

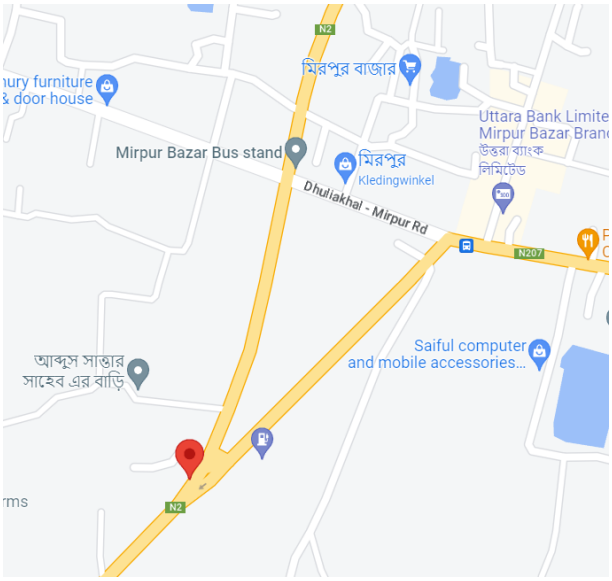

Figure 17:

Figure 18:

Figure 19:


Hunam Kha Enterprise
LPG Auto Gas Station
N208
South Surma Grid
Switching Station...

Figure 20: