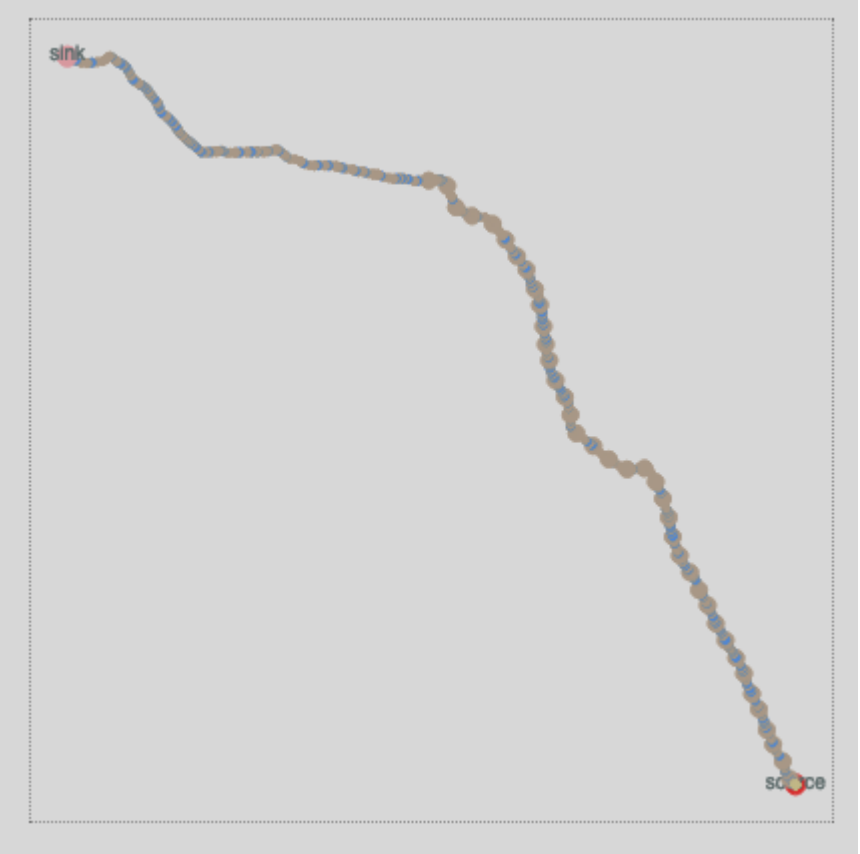
Bangladesh's Transport Infrastructure on the N1:

Component building, Simulation & Scenario Analysis

Group 22
EPA1352

Anna Noteboom	*4564979*
Auriane Técourt	*5397243*
Floris Boendermaker	*4655605*
Job Onkenhout	*4595769*
Zara-Vé van Tetterode	*4577701*



Introduction

This report demonstrates the process of generating a component based Mesa model where we can study the effects of bridge maintenance or unavailability on traffic throughput for the economically important N1 road in Bangladesh, from Chittagong to Dhaka. Modeller bias is reduced by working with a data-driven modelling approach (Keller & Hu, 2019).

Data preparation

The data preparation and cleaning process consists of four main steps and was executed within the *data_processing.py* file. The aim is to merge the roads file and the bridges file adequately, while removing errors and inconsistencies in the dataset. Issues regarding semantics and pragmatics were resolved, for initial data exploration did not point out syntactical errors, as defined by Huang (2013).

Left / L - Removal

Firstly, the bridges file contained duplicate variables. There were *"L'* and *"R"* bridges present at the same locations. The assumption can be made that this is the distinction between the left and right side of the bridges. Since the people in Bangladesh drive on the left side of the road (WorldStandards, 2020) and the modelling will focus on a one-way route from Chittagong to Dhaka, the left (L) bridges were removed from the dataset (We assume that the L/R is determined from Dhaka's perspective, because the N1 'starts' there). This was executed by removing variables with *"(L)"* , *"(left)"*, *"(L)"* and *"(LEFT)"* present in the string within their *name* attribute. This removal did not result in a loss in information in terms of the bridges' important attributes such as *condition or length*. This was verified by semi-manually comparing the left and right data.

Duplicates

The duplicates that were still present after the Left-removal procedure were removed based on their *constructionYear.* Duplicate bridges with an older construction year value than their duplicate were removed. This provided a dataset with no more duplicates and the most recent data.
Assumption: no two LRPS in the same road have the exact same *km* attribute.

Length

To generate consistent lengths, the length data from the BMMS file was combined with the chainage data from the roads file. 1 chainage corresponds to 1000 meters, this was implemented in the data processing. Finally, as the chainage represents the cumulative length of the road, the length of each segment separately was calculated from the chainage in the roads file.

Merging roads and bridges

The bridges and roads file are combined and sorted based on their chainage data. As the N1 road runs all the way from Dhaka to Cox's bazar and the model should only run over the road section between Dhaka and Chittagong, a large part of the road segments and bridges were omitted. To be exact, all road and bridge entries with a chainage of under 241 were removed from the dataset.

Model design

The given model was altered and extended in order to simulate trucks driving over the N1 road. Changes were made in the agents of the model (file name: *components.py)*, the model itself (*model.py*) and the running of the model (*model_run.py)*. A new file *batch_run.py* was added for the experiments.

Changes in the Agent (sub-) classes

The following agents (sub-) classes were changed: Vehicle, Sinks and Bridges. The only change in Vehicles is their speed, which is now set to 48km/h. Sinks now store the time tick at which they remove each truck before removing them from the model: this will be used later to determine the driving times. Bridges underwent four changes. First, their category (A, B, C or D) is now a parameter passed during initialisation. Second, they may now break down based on a probability given by the user when calling the model and dependent on the bridge's condition. Third, they determine a delay time for each arriving truck, based on the bridge's length and a new function named get_delay_time. If the bridge is not broken down, that delay is 0. Lastly, Bridges now save the number of trucks that passed on them and the total delay they caused. This can be used to calculate metrics later on.

Changes in the model itself

The model was changed in various ways. It now takes the N1.csv file as input in order to model the correct road in the correct direction. It also takes four new inputs, which are the probability of the bridges in each category to break down. That probability is then used by the Bridges class to determine which bridges are broken down. Furthermore, four new functions were defined, which act as model reporters. `compute_average_driving` computes the average driving time for all trucks after one run of the model, `compute_worst_bridge` returns the name of the bridge with the highest total delay time, `compute_worst_bridge_delay` returns the average delay time at that bridge, and `get_probs` returns the probabilities of bridges breaking down that were used in the model as an extra way of validation.

BatchRunner

The new file *batchrunner.py* uses mesa's batchrunner to run all scenarios consecutively. Each scenario is run 10 times for 5 days (= 5*24*60 minutes). The probability of bridges of different categories breaking down in each scenario is passed as a list in the batchrunner, and model reporters (the new functions defined in *model.py* and described in the previous paragraph). The results are stored in a DataFrame as one file (*all_scenarios.csv*) as well as in a separate csv file for each scenario.

Experiments and Results

Scenarios 0 through 8 consist of different probabilities of bridges breaking down per condition-category (A, B, C, D), as can be seen in *batchrunner.py*. The outcomes from simulating these scenarios have gone through data analysis and the most important findings have been summarized in various plots. The most insightful observations per plot will be briefly discussed. In general it can be observed that the model portrays the expected behavior: Driving time increases as the probability of bridges breaking down increases.

The dataset contains 127 bridges with condition cat. A, 62 bridges with cat B, 21 bridges with cat. C, and 0 bridges with cat D. The probability of cat. A bridges breaking down therefore has the most influence on the outcomes for the most important KPI, *average driving time.* There are no cat. D bridges present in the dataset, which explains why scenarios 0, 1, and 2 have similar values for

average driving time, because the difference between breakdown-probabilities for cat. D is the only variation between these scenarios (Figure 1).

The higher the breakdown probability, the higher the std. error in the outcomes, meaning estimating individual truck-driving times based on the *average driving times* becomes more difficult as the breakdown-probability increases (Figure 3).

The *average driving time* in scenario 8 can get up to 5 times bigger (+/- 25 hours) than scenario 0 (Figure 2) and has on average a 4 times bigger average driving time (+/- 20 hours) than scenario 0 (Figure 3).

The worst maximum delay of the worst bridge is 142 minutes, active from scenario 4 to 8. The best maximum delay of the worst bridge is approximately 38 minutes for run 6 to 8. This is visualised in Figure 4.

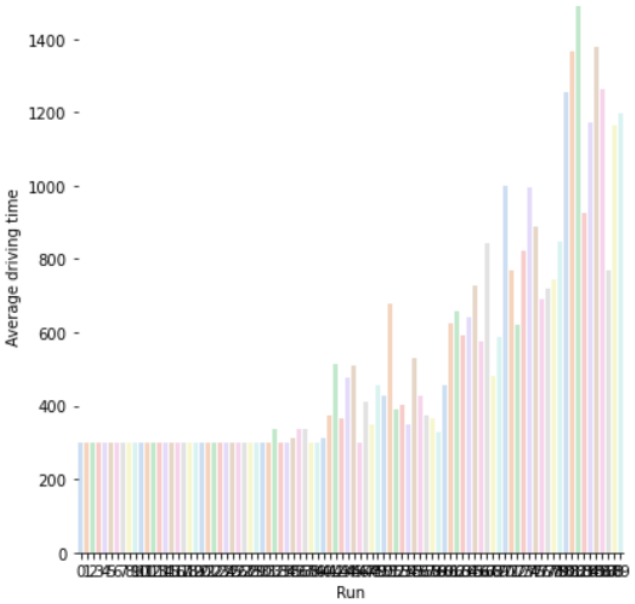


Figure 1:Average driving times

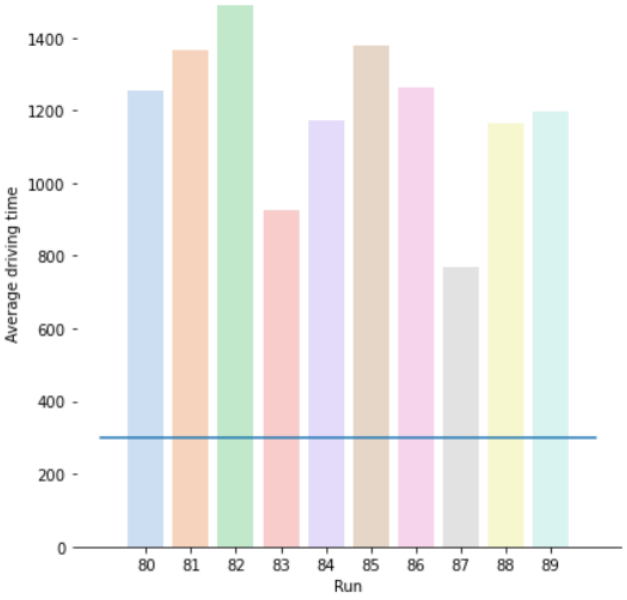


Figure 2: Driving times of scenario 8 and baseline of scenario 0

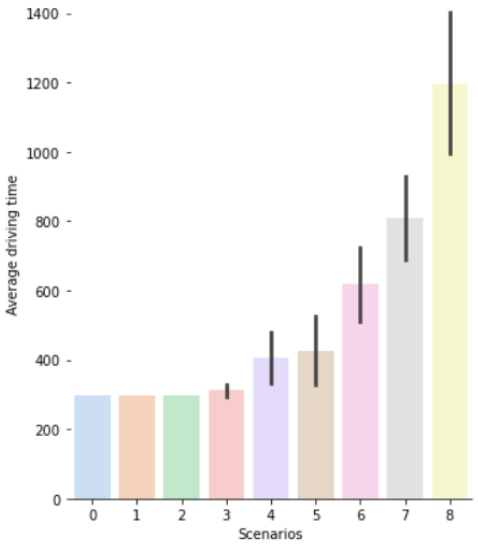


Figure 3: Average driving time per scenario Including standard error

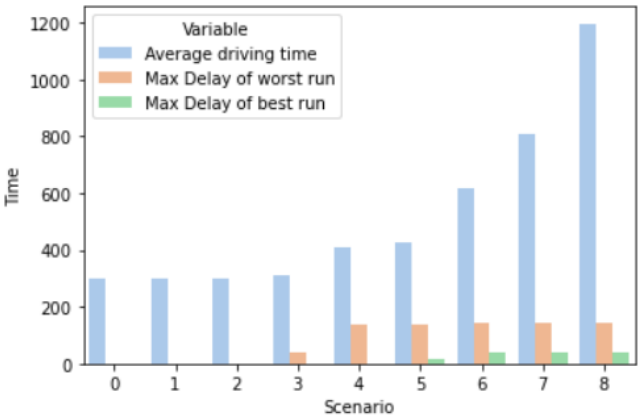


Figure 4: Driving time compared to max delays of the best and worse runs per scenario

Discussion

In order to verify that the delays calculated at each bridge were in line with the assignment, a verification test was done in the *verification.py* file. All details of the test are documented there.

Limitations

There are several aspects of the model that can be improved. Firstly, there are two data-related limitations: not all data of all bridges were present in the BMMS file and a significant part of the data was outdated. A better dataset would result in more reliable model outcomes.
Then, with regards to the data preparation for the model: it is possible too many bridges and roads were omitted from the dataset when cutting off the road at chainage point 241. This could have resulted in the total length of the roads and bridges of only 247 km and the fact that no category D bridges were present on the road. This had implications for the experimental outcomes, because the effect of category D bridges breaking down can not be seen back in the results. Next, the removal of left-side bridges meant that two bridges were removed that did not have a right-side counterpart in the dataset.
Lastly, with regards to the model itself, the implementation of delay times could be implemented in a more sophisticated way to make the model more realistic. For instance, a uniform or triangular delay distribution of broken bridges is relatively coarse. With more research into the distribution of road blockages due to inaccessible bridges, this could be improved. Secondly, the delay could for instance also differ between large and smaller bridges, with bridges that have multiple lanes resulting in less delay time as there is a good chance a portion of the lanes is still accessible whereas a bridge with only one lane, this would be impossible.

Bonus Exercises

Which bridges does Bangladesh need to invest in to decrease lost travel time the best? The scenarios are weighted equally, and scenario 0 and 8 are not taken into account. Overall, the worst breakdowns occur in scenarios 4 to 7. Bridges that perform the worst can be calculated by their highest caused delay time and by the number of times that bridge performed worst in the run. The first interpretation leads to a top 5 of worst bridges as shown in table 1, with delay time describing the longest delay time this bridge caused over the runs. Table 2 shows the top 4 of worst bridges according to the second interpretation. Figure 5 and 6 show the delay time these four bridges caused for the trucks in the runs where they performed worst. In table 2, only 4 bridges are taken into account as all of the other bridges occur either never or only one time as the worst bridge.

Table 1: Bridges with the single worst delays

Ranking	Bridge name	Worst delay time
1	Meghna Bridge	142 min
2	Kanchpur Pc Girder Bridge	141 min
3	Daud Kandi Bridge	140 min
4	Dhoom Ghat Pc Girder	138 min
5	Muddo Sonaisori Bridge	38 min

Table 2: Number of occurrences of worst delay bridges

Ranking	Bridge name	Number of times this bridge caused the highest delay	Delay time in these runs
1	Daud Kandi Bridge	7	+/- 140 min
2	Kanchpur Pc Girder Bridge	5	+/- 140 min
3	Meghna Bridge	5	+/- 141 min
4	Illiot Bazer	2	+/- 38 min

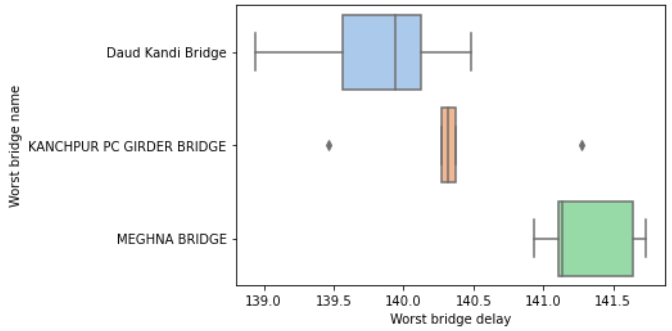


Figure 5: Box plot of worst delay bridges

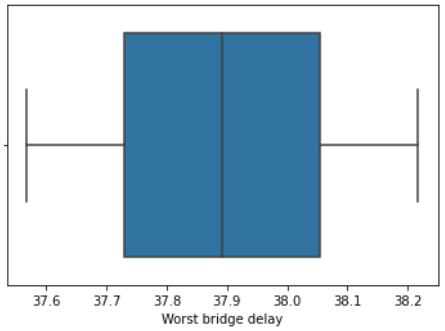


Figure 6: Box plot of Illiot Bazer

All of this taken into account, the Bangladesh government should invest in the Daud Kandi Bridge, Kanchpur Pc Girder Bridge, Meghna Bridge, Dhoom Ghat Pc Girder and the Illiot Bazer Bridge. This way both the amount of breakdowns is contained, as the worst breakdowns in terms of time are contained.

Component design

The model level report functions are defined as separate components. The code is modular, this means that the report functions can be imported in different parts of the model and used in different contexts. This enables others to alter what data should be collected each model run. In this model the following functions are defined to report important model output: `compute_average_driving`, `compute_worst_bridge`, `compute_worst_bridge_delay`, `get_probs`. These functions are imported from the model file into the batch_run file.
This added modularity of model output incorporates the 'reusability' driving force of decomposition and reduces redundancy, which are important criteria for decomposition in simulation modelling (Hofmann, 2004)

References

WorldStandards. (2020). List of left- and right-driving countries. *Worldstandards EU.* https://www.worldstandards.eu/cars/list-of-left-driving-countries/

Keller, N., & Hu, X. (2019). Towards data-driven simulation modeling for mobile agent-based systems. A*CM Transactions on Modeling and Computer Simulation* (*TOMACS), 29*(1), 1-26.

Huang, Y. (2013). Automated Simulation Model Generation, *Delft University of Technology*

Hofmann, M. A. (2004). Criteria for decomposing systems into components in modeling and simulation: Lessons learned with military simulations. *Simulation, 80*(7-8), 357-365.