Assignment 2
Building Components for Data-Driven Simulation
Advanced Simulation

William Horeman (4726669),
Nils Groenhof (5092108),
Ameera Raissa Shafa (6047572),
Berend Domhof (4659724),
Tobias Schmeink (4954270)

March 7, 2025

Contents

1 Introduction

This assignment develops an agent-based model (ABM) using MESA to simulate truck movement along Bangladesh's N1 highway from Chittagong to Dhaka. The model assesses how bridge failures—caused by floods, earthquakes, landslides, or other disasters—impact truck delays.
To analyze these effects, we use multiple scenarios. Scenario 0 represents normal conditions with no bridge failures. Scenarios 1-9 introduce different failure probabilities across bridge categories (A–D), where A has the lowest and D the highest collapse risk. The breakdown probabilities for each scenario are summarized in Table 1.

Scenario	Cat A (%)	Cat B (%)	Cat C (%)	Cat D (%)
0	0	0	0	0
1	0	0	0	5
2	0	0	0	10
3	0	0	5	10
4	0	0	10	20
5	0	5	10	20
6	0	10	20	40
7	5	10	20	40
8	10	20	40	80

Table 1: Scenario data with breakdown probabilities

Bridge failures are determined at initialization, meaning delays remain constant throughout the simulation—no repairs occur. Since trucks follow similar routes and encounter the same bridge delays, variations in delay time are considered unrealistic, ensuring that all trucks experience consistent travel disruptions.

Bridge Length	**Delay Time for a Truck**
Over 200 m	Triangular(1, 2, 4) hours
Between 50 and 200 m	Uniform(45, 90) minutes
Between 10 and 50 m	Uniform(15, 60) minutes
Under 10 m	Uniform(10, 20) minutes

Table 2: Bridge length and corresponding delay time for a truck based on the length of the road

2 Data Preparation and Model Explanation

2.1 Data Preparation

This simulation aims to evaluate the impact of potential bridge failures on delay times of vehicles driving along the N1 route between Chittagong and Dhaka in Bangladesh. To prepare the required data, a CSV file needs to be created, listing all the nodes along the N1 route arranged in the order of their chainage.

The model takes input data with the following columns: road, id, model type, name, lat, lon, length. The roads column is the indicator which road the objects are located on. The ID is an unique ID assigned to each object to identify them. The model type is indicated where an object is a link (a piece of road) or a bridge. The name simply indicates more specifically what kind of object it is. Lat and lon are the latitude and longitude of each object; the coordinates. Lastly there is the length, which indicates the length of each object. Below we will describe all the transformations that were done with the datasets to have them in a usable format for the model.

2.1.1 Bridge

- Bridges along the N1 route are extracted from the BMMS_overview.xlsx file. The data is filtered to keep only bridges associated with road N1.
- The column type is renamed to model_type to align with the roads dataset.
- Additional necessary columns, such as length, are calculated based on the difference in chainage values.
- Redundant columns such as lrp and gap are removed to maintain consistency with the roads dataset.
- The final dataset contains the following columns: road, name, chainage, condition, model_type, length, lat, lon; these are the right columns for later use in the model.

2.1.2 Road

- The road dataset is loaded from _roads3.csv and filtered to retain only road N1 data.
- The length column is computed as the absolute difference in chainage values to maintain correct road segment distances.
- The type column is standardized to lowercase and updated to classify elements correctly: Entries matching known bridge types (e.g., "culvert", "bridge") are labeled as bridge. Other entries are labeled as link.
- The first and last row are explicitly set as source and sink, representing the starting and ending points of the route.
- The type column is then renamed to model_type, since this is the correct column name used in the model.
- Any unnecessary columns, such as lrp and gap, are removed.

2.2 Data Processing and Experimenting

2.2.1 Data Structure

- Bridges are categorized by their condition, allowing analysis of how structural integrity impacts traffic flow.
- Bridges and roads are combined into a single roads DataFrame to create a unified infrastructure model.
- The dataset is sorted by chainage to maintain sequential order along the route. The two datasets were merged based on chainage, since this is the variable that determines the order of the object on the road.
- Each entry is assigned a unique id, starting from 1000,000 to ensure consistency in reference numbers.
- Missing condition values for bridges are filled with a default value of "A", since this is by far the most common value of condition.
- Duplicate chainage values are detected and resolved by prioritizing bridges with "(L)" in their name. In the data there were many bridges with either both L and R or double bridges with L and R. This is the case since certain bridges had different bridges for the left and right lanes (This was verified using Google Maps). Also, certain bridges had L or R meaning that they are a bridge connecting to the main road; thus, they shouldn't count as a bridge in the N1 road. Because of time constraints, all L and R bridges were assigned to their L bridges.
- The length column is recalculated using the increase in chainage per step. It is important this should be done after the merging of the two datasets, since otherwise this would give a distorted length value.

2.2.2 Model Structure

The model is structured into four parts:

- Components.py is used to craft agents. There are five important agents: Link, Bridge, Truck, Source, and Sink. Link has no specific behaviour, Bridge could collapse, Trucks could drive or wait, Source generates truck, and Sink removes trucks. Most modification to the code took place in here, like assigning delays.
- Model.py serves as the core model representing a transportation network within Bangladesh. Its main purpose is to mimic the dynamics of vehicle movements in the network using different scenarios. During each step, it updates the vehicles positions and velocities, considering the conditions of the vehicles and infrastructure.
- Model_run.py operates as a simulation module 'Bangladesh Model' to perform the simulation over a designated period of time (through ticks) and generate output that we can immediately review in the terminal.
- Model_viz.py includes scripts to visualize the model.

3 Methodology

This section encompasses the updates we did to the model.py and components.py files. The changes we did aims to refine the simulation's representation of traffic and infrastructure interactions.

3.1 Missing methods/steps in the original file

In the original file the scenarios, delay times and datacollection methods were missing. There was also no method present to save the output data. This meant that only 0 could be ran (assuming the data has been fully cleaned). By including new methods in model.py and components.py, one could simulate the scenarios as provided in the assignment. These methods are described below.

3.2 New methods added to the model

The following methods were added in model.py:

- **determine_broken_bridges**: Identifies broken bridges at the start of the simulation based on their condition and breakdown probability.
- **get_average_driving_time**: Computes the average driving time of all trucks that have completed their journey. The breakdown probabilities for each scenario was added in the initialization of the BangladeshModel class.

The following methods were added in components.py:

- **assign_delay** returns a delay in minutes based on length. For lengths under 200, it uses a uniform distribution: (10–20 min for ¡10), (15–60 min for 10–50), and (45–90 min for 50–200). For lengths greater than 200, it applies a triangular distribution (60–120 min, peak at 240), favouring higher delays.
- **get_delay** returns the delay time if the bridge is broken, otherwise it returns 0.
- **get_delay_time** returns get_delay.

  Note that there are also many print statements present in the code to check if the code works correctly, which were used for debugging purposes. These statements were left in for understandability of the use.

3.3 Modified methods

Some methods had to be modified, which often involved the initialization of certain methods. For instance, the property of Condition was not present in the code. To make sure that the condition was correctly integrated, some methods had to be modified.

4 Results

4.1 Scenario Results

In this simulation, each scenario was repeated ten times using random seeds to measure truck travel time from Chittagong to Dhaka. The bar plot below represents the average travel time for all trucks in each scenario.

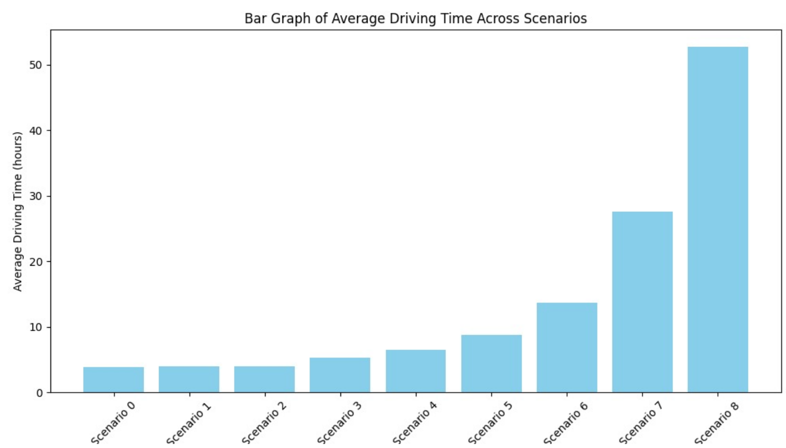

Figure 1: Barplot Average Driving Time per Scenario

The figure shows that as scenarios progress, higher breakdown probabilities lead to more delay time, making vehicles need to wait. Ultimately, this increases the average driving time.

The average driving time for Scenario 0 is the lowest since, in this scenario, there are no bridges breaking down. Scenario 1 and Scenario 2 show a slight increase in the average driving time, perhaps because the breakdown probability occurs only for bridges under condition 'D' with a percentage of 5% and 10%, respectively. With only two such bridges present, the increase in probability has a minor impact on the driving time, hence the small increase shown in the figure.

Scenario 3 showed a more visible increase as bridges under condition 'C' now have a breakdown probability of 5% while bridges under condition 'D' remain at 10%. Having long bridges (over 200m) present leads to significant delays, affecting the driving times more noticeably due to their longer delay times. Similarly, Scenario 4 continues the increasing trend in driving time as the breakdown probabilities increase twice from the Scenario 3.

Scenario 5 introduces the breakdown probability of bridges under condition 'B.' Bridges under this condition vary in size, from under 50m to longer than 200m. Despite the low breakdown probability in this scenario, the large number of bridges under this condition leads to a noticeable impact on driving times. Scenario 6 presents quite a gap compared with Scenario 5. In this scenario, the combined effect of more frequent breakdowns of the long bridges substantially contributes to the increased driving time and variability.

Scenario 7 showed further increase in the average driving time. In this scenario, bridges under the condition 'A' that are numerous is now introduced to have a breakdown probability of 5%. Alongside the existing breakdown probabilities of the other conditions, the inclusion of this adds much to the overall increase in driving times due to their massive occurrence despite their short delay time. Lastly, Scenario 8 presents the highest average, reflecting significant delays. All bridge conditions now have elevated breakdown probabilities, with A at 10%, B at 20%, C at 40%, and D at 80%. The high likelihood of delays, particularly for the five longer bridges in conditions B and C, contributes much to the high driving time.

Overall, although the theoretical travel time (given a distance of 460 km and an expected speed of 48 km/h) would be approximately 9.6 hours, the simulation effectively captures the expected trend, illustrating the substantial impact of bridge reliability on overall driving efficiency.

4.2 Bonus Question

Try to find out which 5 bridges on the N1 the Bangladesh Government should invest in to decrease lost travel time best, when we weigh scenarios 1 though 7 equally.

When inspecting the dataset, the bridges which result in the most severe delays can be easily found. When looking at the breakdown probabilities, one can observe that the bridges with condition 'D' breakdown most easily (see Table 1. As bridges with a longer length result in more severe delays (see Table 2), one should look for long bridges with condition 'D'. These bridges are:

- The **Modonhat Bridge**. This bridge has a length of 737 meters with condition 'D'.
- The **Morichia Bridge**. This bridge is 447 meters long with condition 'D'.
- The **Shapur Steel Beam and RCC Slab Bridge**. This bridge is 433 meters long with condition 'D'.
- The **Wheke Kang Bridge**. This bridge is 228 meters long and has condition 'D'.
- The **Donmia Bridge**. This bridge is 202 meters long and has condition 'D'.

What model and model component design (and/or improvement) did you do with respect to, e.g., modularity and cohesion? Specify and discuss those with the rationale.

The modifications to the model improve modularity and cohesion, aligning with established software engineering principles (Hofmann 2004). New methods were namely created in locations (notebooks) which contained methods with a relatively high cohesion to the new methods. It for instance made sense to calculate the delay time in the bridge class, as a delay time could be a property of a bridge.

What could have been enhanced is that the scenarios should have been defined in model_run, as input parameters. This way the code would have been more object oriented.

5 Discussion, Conclusion, Reflection

The results do not align with the average 6 hours for the 287 km which was provided. Despite much effort, the cause for this was not found. The does however give an impression on how scenario's could impact the delay time, which seem to occur quadratic. Based on the results, bottlenecks in the route were identified, described in the bonus question.

References

Hofmann, Marko A. (2004). "Criteria for Decomposing Systems Into Components in Modeling and Simulation: Lessons Learned with Military Simulations". In: *SIMULATION* 80.7–8, pp. 357–365. DOI: 10.1177/0037549704049876.

6 Appendix A: Acknowledgement

6.1 AI Statement

In accordance with the guiding principles for the responsible use of AI tools, as outlined by TU Delft, the use of generative AI tools such as ChatGPT in this analysis has been conducted responsibly and, hereby, transparently.

AI was used to get the model to do what we wanted. This meant that the ChatGPT was used to generate functions which could be applied on the code. It was checked whether the suggested function by ChatGPT resulted in the behaviour that we wanted to see according to our own mental models. Specifically in debugging, ChatGPT was valuable, as this software could direct us to the bugs present in our software.

6.2 Task Division

Table 3: Distribution of the Workload

Student	**Task**
Ameera	Work on the code for the datacollection and trucks, report writing
Nils	Work on the code, report writing, report writing
William	Worked on the code for the scenarios and bridges, report writing
Tobias	Cleaned the data, report writing
Berend	Cleaned the data, report writing