Building Components for Data-Driven Simulation

ADVANCED SIMULATION

EPA133a

Authors:

Bettin Lorenzo (6132928)
Dell'Orto Alessandro (6129161)
Le Grand Tangui (6172075)
Precup Ada (5240719)
van Engelen Ralph (4964748)

March 7, 2025

TU Delft Delft University of Technology

Contents

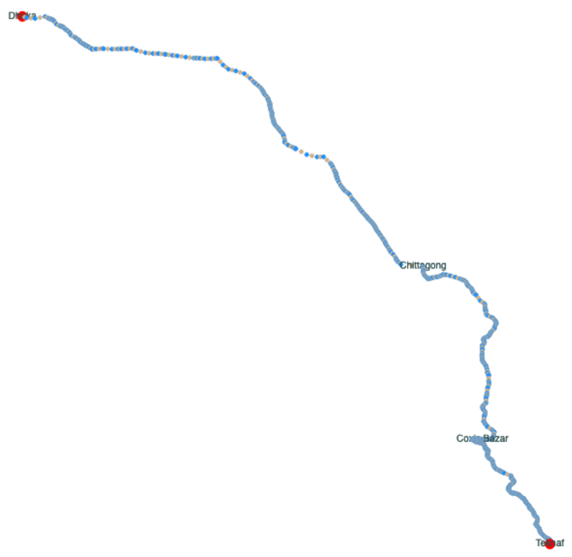
Introduction

Assignment 2 of the Advanced Simulation course focuses on modeling Bangladesh's road network by constructing its key components, such as vehicles, roads, and bridges, and simulating their interactions within the traffic system.

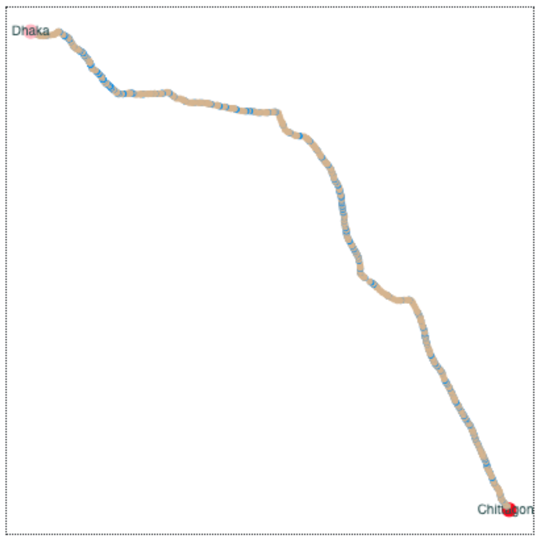
The system follows Component-Based Modeling (Hofmann, 2004), defining essential entities like sources, sinks, links, bridges, and vehicles. This modular structure enhances transparency, facilitates parallel development, and improves adaptability for future modifications.

Building on the course's foundational work, we introduce maintenance attributes for bridges, influencing traffic flow by causing delays and altering travel times. By simulating varied bridge conditions, we examine their effects on congestion and infrastructure efficiency.

Our simulations focus on National Road N1, Bangladesh's primary transportation corridor linking Dhaka to Chittagong (Figure 1). As a vital economic route, disruptions on this highway impact national socio-economic stability (DBpedia Contributors, 2024). Modeling different maintenance scenarios provides insights into traffic management strategies and infrastructure planning.



(a) The entire N1 road.



(b) The section of the N1 highway connecting Dhaka and Chittagong. Traffic flows from the port city, Chittagong, to the capital, Dhaka.

Figure 1: Relationship between bridge breakdown and traffic.

1 Methodology

1.1 Updating the road dataset

The model input dataframe needed to be updated with components from roads. First, because the road data overlapped with the data of the bridges the overlapping coordinates needed to be deleted from the dataset. This was necessary to ensure that bridges were treated as distinct elements rather than being duplicated within the road network. This was done by converting the bridge LRP values into a set and then using this set to filter out the bridge coordinates from the road dataset. Second, because only the longtitude and latitude was given for the road data, the distances between points were not yet known. This distance was constructed by creating links between LRPs. Each link was defined by the coordinates of its starting LRP. The haversine function was then used to calculate the lenth of each link in meters. Each link was then named based on the name column of the starting LRP of the link.

1.2 Updating the bridge dataset

The input dataframe then needed to be updated with bridge components. Since the bridges that were deleted earlier as overlapping coordinates from the dataframe needed to be reincorporated, they were now reintroduced explicitly as bridges. The challenge with this was determining the exact location where each bridge should be insterted into the dataset so that it connects to a road. To solve this, first, a function was made that finds the shortest distance between a bridge and a link so that the closest LRPs that correspond to the bridge based on minimum Euclidean distance could be found.
The bridges could then be incorporated back into the dataset as new rows. However, since bridges had to be inserted at specific points within the dataset this had to be done in the correct order. This was done by sorting the bridges based on their insertion indices in descending order before adding them to the dataframe. Left and right bridges were kept since a disruption in one of the two could result in a disruption on the other bridge as well, as traffic would have to be rerouted.

1.3 Formatting the input file

The final step involved formatting the dataset to use in modeling. This was done by assigning a unique ID to each row, starting from a predefined value. This ensured that each row and bridge component can be uniquely identified. Moreover, the first row of the dataset was labeled as the source and the last row as the sink to indicate the entry and exit points of the road network.

1.4 Modifications to `model.py`

The `model.py` script, where the `BangladeshModel` class is implemented, has been modified to ensure proper usage of the updated input CSV file and to fulfill the simulation requirements:

- **Breakdown Probabilities:** An input parameter was added to the `BangladeshModel` class, providing the model with the failure probability for each class. These probabilities vary for each scenario. The parameter is given as a dictionary.
- **Output Variables:** Two attributes, `travel_times` and `broken_bridges`, were added to the class. These attributes store:
  - `travel_times`: A list collecting the travel times (in ticks) for each truck that reaches its destination.
  - `broken_bridges`: A list tracking the number of bridges that collapse during the simulation.
- **Input Data File:** The model now reads data from the updated dataset `final_input_data.csv`.
- **Bridge Object:** When creating a `Bridge` agent, an additional parameter is passed, indicating the quality category of the bridge.

1.5 Modifications to `components.py`

Enhancements in the Bridge Class

The `Bridge` class now incorporates several key updates aimed at reflecting a more realistic behavior:

- **Breakdown Probabilities:** In the constructor, a new dictionary named `breakdown_probabilities` is introduced. This dictionary maps various quality categories (e.g., 'A', 'B', 'C', 'D', 'Z') to probabilities indicating the likelihood of the bridge breaking down. For instance, a low-quality bridge (category 'D') has a 20% break-down chance. This addition allows the simulation to account for different levels of infrastructure reliability.
- **Revised Delay Time Calculation:** The `get_delay_time` method has been overhauled to compute delays more realistically. Instead of returning a placeholder random delay, the method now:
  - Checks whether the bridge is broken by invoking a new method (`is_broken`).
  - Determines delay based on the bridge's length:

    * For bridges longer than 200 meters, a delay is sampled from a triangular distribution (with values in hours converted to minutes).
    * For bridges between 50 and 200 meters, a uniform distribution between 45 and 90 minutes is used.
    * For bridges between 10 and 50 meters, a uniform delay between 15 and 60 minutes is applied.
    * For very short bridges (10 meters or less), the delay is sampled uniformly between 10 and 20 minutes.
  – If the bridge is not broken, the method returns a delay of 0.

- **Introduction of the `is_broken` Method:** This method retrieves the breakdown probability for the given quality category and compares a random value against this probability to return either `True` (if the bridge is considered broken) or `False`.

These modifications in the `Bridge` class help ensure that the delay experienced by vehicles is not arbitrary but closely linked to the bridge's condition and physical attributes.

Enhancements in the Vehicle Class

The `Vehicle` class now includes improvements that make vehicle behavior more dynamic and measurable:

- **Travel Time Tracking:** A new attribute, `travel_time`, has been added to the `Vehicle` class. This attribute is:
  – Initialized when a vehicle is created.
  – Incremented in every step (tick) of the simulation.
  – Recorded in the model's `travel_times` list when the vehicle reaches a sink (i.e., is removed from the simulation).

  This change allows for the collection of performance metrics that can later be analyzed to assess the overall efficiency and timing within the simulation.

1.6 Modifications to `model_run.py`

The `model_run.py` script now performs the expected simulation:

- A pandas DataFrame is created to include all the different breakdown probabilities for the various bridge categories and for each scenario.
- Runtime and seed values are assigned. A set of 10 different seeds is defined to perform 10 different iterations.
- A list called `results` is defined to store all relevant simulation outcomes.
- For each scenario, the script runs the simulation for the specified number of ticks (minutes) with the corresponding breakdown probabilities and stores the average travel time, the number of trucks that arrived at the sink, and the number of bridges that broke during the simulation.
- Finally, the script stores all output data for each scenario in a CSV table, and creates another to store average values among the 10 iterations.

2 Results and Discussion

Table 1 presents nine simulation scenarios, each characterized by its average travel time (in minutes), the total number of trucks processed, and the number of broken bridges. All the shown values are averaged over 10 iterations with different seeds.

- **Scenario 0:** This scenario serves as a reference point where no infrastructure failures occur, allowing for optimal network performance. The network operates without disruptions, and all vehicles follow their expected routes without delays caused by broken bridges.

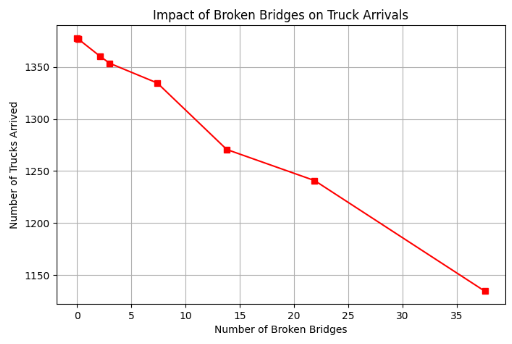
Scenario	Average travel time (minutes)	Number of trucks	Number of broken bridges
Scenario 0	312	1378	0
Scenario 1	315	1377	0.1
Scenario 2	315	1377	0.1
Scenario 3	362	1360	2.1
Scenario 4	386	1353	3.0
Scenario 5	421	1334	7.4
Scenario 6	579	1270	8
Scenario 7	648	1240	21.9
Scenario 8	1018	1134	37.6

Table 1: Summary of Scenarios

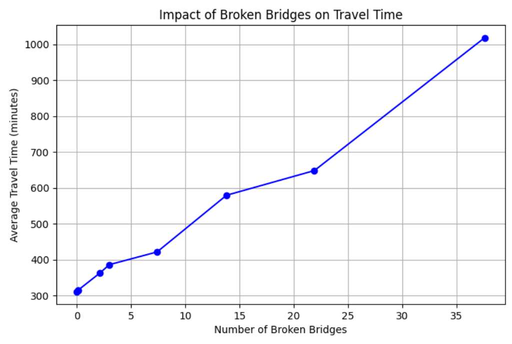
- **Scenario 1:** In most of the iterations the outcomes are identical to Scenario 0, even if D quality bridges are now assigned a probability of failure. This highlights the importance of running simulations multiple times.
- **Scenario 2:** In this case, iterations were again almost without any breakage, resulting in the same average output data, suggesting the number of iterations could be enhanced.
- **Scenario 3:** Now, multiple bridge failures cause a moderate increase in travel time and a noticeable drop in the number of trucks completing their trips, demonstrating the compounding effects of failures.
- **Scenario 4:** As infrastructure reliability declines further, vehicles experience longer travel times and fewer trucks manage to complete their journeys.
- **Scenario 5:** The number of broken bridges nearly doubles, leading to significant congestion and rerouting delays, along with a more pronounced drop in truck throughput.
- **Scenario 6:** The continued increase in failures results in a persistent rise in travel times, indicating that as reliability decreases, the system's capacity to manage vehicles deteriorates further.
- **Scenario 7:** With an escalating number of failures, the network becomes heavily compromised, resulting in a substantial decrease in truck throughput and increased travel times.
- **Scenario 8:** The extreme case of widespread bridge failures drastically increases travel times and significantly reduces the number of trucks that complete their journeys.

2.1 Scenario Conclusions

Figure 2, as it follows, illustrates the relationship between the number of broken bridges and two key performance indicators: the number of trucks that successfully arrive (Figure 2a) and the average travel time (Figure 2b). In Figure 2a, the red line shows a nearly linear decline in the number of trucks as broken bridges increase. This steady downward trend indicates that each additional bridge failure imposes further constraints on the network, thereby reducing its capacity to accommodate traffic. Although the slope may not be perfectly uniform across the entire range, it remains broadly linear, particularly at lower to moderate levels of bridge failures. On the other hand, Figure 2b shows how average travel time grows in relation to the number of broken bridges. In the early stages, when fewer bridges fail, the increase in travel time appears roughly linear. Each additional broken bridge contributes a predictable increment to the total delay experienced by vehicles. However, once the number of failures becomes sufficiently large, the slope becomes steeper, suggesting that the system enters a more congested state. In these higher-failure scenarios, vehicles experience significantly longer rerouting or waiting times, causing a sharper rise that exceeds a simple linear pattern. Overall, the results confirm a close-to-linear relationship in both declining truck arrivals and increasing travel times for lower to moderate levels of bridge breakdowns. Moreover, by changing the simulation seeds, the results might vary, and it would be interesting to see if these trends persist under different random conditions. As failures grow more extensive, however, the network's performance deteriorates more dramatically, highlighting that beyond a certain threshold, the impact of each additional broken bridge compounds, leading to disproportionately higher delays.



(a) Illustration of the relationship between the number of broken bridges and vehicles crossing N1. The relationship is described based on the average travel time and number of trucks obtained from different seeds.



(b) Illustration of the relationship between the number of broken bridges and average travel time.The relationship is described based on the average travel time and number of trucks obtained from different seeds.

Figure 2: Relationship between bridge breakdown and traffic.

2.2 Assignment Limitations

2.2.1 Physical Limitations

The model assumes constant speed and congestion, neglecting terrain, road conditions, and traffic variations. Vehicle routes, starting points, and destinations remain fixed, omitting potential driver detours.

Bridge breakdowns occur stochastically at each tick rather than progressively worsening over time, which does not fully capture real-world degradation. Additionally, assigned breakdown probabilities are arbitrary and not entirely representative of reality.

2.2.2 Structural Limitations

The chosen decomposition method introduces limitations. Decomposition, as defined by Hofmann, 2004, breaks complex systems into manageable parts. Our code is structured into Python scripts by class and method, but our level of abstraction may affect model accuracy. For instance, we use a single "vehicle" class instead of differentiating heavy, medium, and light-duty vehicles, despite differences in speed, load, and routing behavior. Similarly, roads and bridges could be further decomposed.

Another limitation is the absence of behavioral modeling. The model assumes all drivers react uniformly to disruptions, such as bridge failures, when responses likely vary. Additionally, the driving patterns and attitudes in Bangladesh may differ from assumptions made by a European team, highlighting the need for behavioral analysis.

2.3 Future Recommendations

Future improvements could focus on parametric and structural refinements. A parametric approach would enhance input accuracy by incorporating variables like traffic and topography. Expanding the model to include additional roads would further improve realism.

A structural approach would involve adjusting decomposition levels, such as introducing distinct vehicle classes to better reflect diverse traffic conditions. Behavioral analysis could refine model accuracy, making simulations more realistic. Furthermore, Keller and Hu, 2019 highlights the bias in knowledge-driven models like ours. Transitioning to a data-driven approach could mitigate this, though challenges may arise due to limited training data.

Finally, increasing the number of iterations would provide a broader result set. While two scenarios produced identical output statistics, the simulation still identified general trends.

3 Acknowledgments

3.1 Task Division

Given the modularity with which the model was built, during this assignment we prioritized that every team mate contributes to the code. To this end, we created different teams that tackled the simulation.

- First Team: Alessandro & Tangui; they focused on developing a method to introduce and diversify the probabilities of a bridge breaking into the model.
- Second Team: Alessandro & Lorenzo; they focused on defining the delays introduced by a broken down bridge, as well as determining a method to track and store these delays.
- Third Team: Ada & Ralph; they focused on constructing the input-file for the model based on real-world data from the Bangladesh road network.

The team collaborated on writing the report, each pair of partners detailing on the section most related to their task in the assignment.

3.2 AI Acknowledgment

Given the different experiences the team members have with coding, for this assignment, heavy guidance was needed from ChatGPT to explore the functions of Mesa, for example, learning how to extract the vehicle's travel time from the model run. We also used large language models (LLMs) to understand the parts of the code we didn't interact with too much, and it helped us develop a clear, big-picture understanding of the code.

ChatGPT was also instrumental in problem-solving git issues throughout the assignment and helped explain why the terminal was not accepting commits sometimes.

LLMs were also employed to help structure the markdown files, explain the code (README.md), and understand markdown syntax. The approach was to attempt to explain a piece of code in the README and have ChatGPT help format it.

ChatGPT was also useful in formatting the present report, as it helped with syntax for introducing figures and citations in the latex code.

References

DBpedia Contributors. (2024). N1 (bangladesh) [Accessed: 2024-03-05]. https://dbpedia.org/page/N1_(Bangladesh)

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. https://doi.org/10.1177/0037549704049876

Keller, N., & Hu, X. (2019). Towards data-driven simulation modeling for mobile agent-based systems. *ACM Trans. Model. Comput. Simul.*, *29*(1). https://doi.org/10.1145/3289229