Assignment 2

Building Components for Data-Driven Simulation

EPA133a Advanced Simulation

Group 16

Rachel Delvin Sutiono, No. 6284736

Celia Martínez Sillero, No. 6222102

Daniela Ríos Mora, No. 6275486

Thunchanok Phutthaphaiboon, No. 6141153

Yao Wang, No. 6157513

Outline

1. Introduction

This assignment focuses on simulating vehicle travel times under different infrastructure conditions in Bangladesh. Using Mesa library, an agent-based modeling tool, we built a simulation model to study how bridge conditions impact driving time on the N1 road.

To achieve this, we first prepared and generated infrastructure data, which is explained in Chapter 2. We then modified the simulation model to include bridge failure scenarios, described in Chapter 3. Finally, we analyzed different scenarios to assess how bridge breakdowns affect travel time in Chapter 4. Lastly, the bonus exercises explore which bridges should be prioritized for investment and the reflection on component building in Chapter 5.

2. Generation of the demo data

The data preparation process aims to clean and combine bridge and road data from different sources (*BMMS_overview.xlsx* and *_roads3.csv*) and create a demo dataset that can be used for the simulation. The final dataset should have a specific structure like the example demo file and include necessary additional attributes for the simulation, such as the bridge's conditions. The source code for this process can be found in "*EPA133a-G16-A2\notebook\CreateDemoFile.ipynb*". Steps taken are as follows:

1. **Load data:** Loading the bridge and road data RMMS_overview.xlsx and _road3.csv is the first step to access the raw data needed for the simulation.
2. **Remove duplicates from bridge and road data:** Creating a road_lrp column helps uniquely identify each bridge entry. Cleaning the name column and removing duplicates ensures data consistency and accuracy. This step eliminates redundant entries and ensures that each bridge is uniquely identified.
3. **Combine bridge and road data:** Renaming columns and adding a model_type column standardizes the data format. Merging the cleaned bridge and road DataFrames into df_concat and sorting it ensures a comprehensive and organized dataset. This step combines all relevant data into a single, demo-like DataFrame, making it suitable to use in the simulation.
4. **Build and save the demo DataFrame:** Filtering out rows with NaN condition in model_type as bridge ensures data completeness. Creating a new DataFrame with the specified structure and calculating length provides the necessary attributes for the simulation. This step creates a demo dataset that meets the simulation's requirements (EPA133a-G16-A2\data\processed\demo_100.csv).

The result of this cleaning process is visualized in **Figure 2.1.** There are 628 bridges across the whole N1, including culverts. Their conditions and count can be seen on **Annex A**.

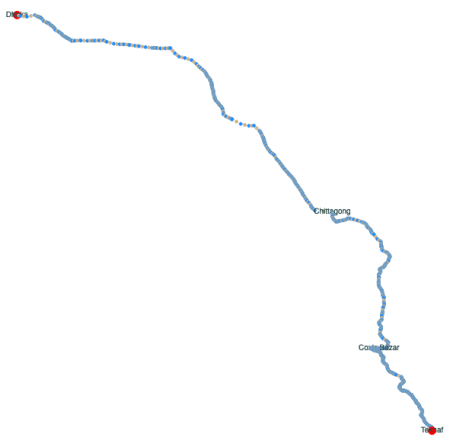

Figure 2.1: Road N1 from Chittagong to Dhaka

3. Modification of the simulation model

Modifications are done to the former model. This also includes the addition of new attributes and methods necessary to incorporate broken bridges, its consequent delay, and vehicle driving time data collection. The codes are—already—structured to accommodate modularity and cohesion in the models. Below are the descriptions of what and how modification and addition to model (‘*model.py*’), components (‘*components.py*'), and simulation run (‘*model_run.py*’) contribute to the simulation’s functionality.

3.1. model.py

3.1.1. BangladeshModel(Model)

Attributes

1. ‘*self.driving_times*’ is a list storing the driving time of all vehicles that successfully reach a sink; useful for computing average driving time at the end of simulations.
2. ‘*self.bridge_delays*’ is a dictionary that records the total delay time for each broken bridge (key: bridge_id; value: cumulative delay time for all vehicles); it is useful for identifying bridges and their delays.
3. ‘*self.total_wait_time*’ tracks the total time all vehicles spent waiting due to broken bridges; useful for calculating the average delay time per vehicle.
4. ‘*self.probabilities*’ stores the values that define the likelihood of bridges breaking down passed from ‘model_run.py’ which enables different scenarios for analysis.
5. ‘*self.scenario*’ stores the current scenario number and links it to the corresponding probability set.
6. ‘*self.broken_bridges*’ stores IDs of broken bridges at the start of the run, ensuring all vehicles face the same set of broken bridges in that simulation run.

Methods

1. ‘*generate_model*’ which reads a CSV file containing infrastructure components is slightly modified.
    a. It’s adjusted to read the new ‘id’ format, which contains the road and id numbers, separated by an underscore (RoadName_IDNumber).
    b. The new attribute, ‘condition’, is included when creating the Bridge agent.
2. ‘*determine_broken_bridges*’ is added to identify which bridges are broken at the start of the simulation, ensuring all vehicles in that run experience the same broken bridges. It uses

predefined probabilities for each condition (A, B, C, D) to decide breakage and returns a set of broken bridge IDs.

3. '*get_average_driving_time*' computes average travel time for vehicles that reach the sink.
4. '*get_total_delay_time*' returns the total wait time due to broken bridges.
5. '*get_average_delay_time*' computes the average wait time per vehicle.
6. '*get_broken_bridges*' returns a list of broken bridge IDs.

3.2. components.py

3.2.1. Infra(Agent), Infrastructure Base Class

Attributes

1. '*self.length = length * 1000*' converts lengths from kilometers to meters, ensuring consistent unit calculations.

3.2.2. Bridge(Infra)

Attributes

1. '*self.probabilities = model.probabilities*' gets probability values from BangladeshModel.

Method

1. '*get_delay_time*' determines random delay times—different random distribution for different length categories—for vehicles crossing broken bridges. Longer bridges cause more delay time when broken.

3.2.3. Vehicle(Infra)

Speed is now set to 48 km/h as the assignment required.

Method

1. '*drive_to_next*' which manages vehicle moves and waiting behavior when reaching a new infrastructure, is now being added a logic allowing vehicles to wait at broken bridges and track their delay time.
    a. Added a *while True* loop replacing the former recursion logic to avoid stack overflow and iteratively move through multiple infrastructure elements in a single function call.
    b. Delay time calculations only happen when the vehicle first reaches a broken bridge. If the bridge is broken, It checks if '*self.waiting_time == 0*' before assigning a delay to prevent overwrites, then computes and assigns a random wait time.

```
if next_infra.unique_id in self.model.broken_bridges:
        if self.waiting_time == 0:
                self.waiting_time = next_infra.get_delay_time()
```

c. Added a bridge delay accumulation function which tracks delays per bridge, ensuring that each broken bridge's delay is only recorded once per vehicle. It updates and stores delays in '*self.model.bridge_delays*'.

```
if next_infra.unique_id not in self.model.bridge_delays:
        self.model.bridge_delays[next_infra.unique_id] = 0
self.model.bridge_delays[next_infra.unique_id]+=self.waiting_ti
me
```

3.3. model_run.py

Modifications

This is where the simulation execution is run. Each simulation runs for 7,200 ticks representing 5 days, as 1 tick is assumed to be equal to 1 minute. A '*scenario*' dictionary is defined separately to ease performing different scenario analysis. Currently, 9 different scenarios, each specifying the probability of failure for bridge conditions A, B, C, and D are defined—as per required. Higher scenario values represent increasing bridge breaking rates. Each scenario is run 10 times with different random seeds.

After running each scenario, it stores results, road name, scenario, seed, average driving time, total waiting time, average waiting time, and broken bridges list, in '*data_list*' to avoid overwriting previous runs. Data list is converted to a DataFrame and is saved as a CSV file after all seeds for a scenario are completed. At the end, each scenario has a separate CSV file with multiple seeds.

Simulation Execution

To reduce computational cost or running time, disabling all printing functions in each '*step*' method is suggested. Simulating all 9 scenarios in one run, each with 10 replications, while only enabling '*Sink*' and '*Source*' printing functions, took 15 minutes. The result can be seen in Annex B. This was done using a Macbook Pro with specifications as follow:

- Processor : 1,4 GHz Quad-Core Intel Core i5
- macOS : Sonoma 14.6.1
- Memory : 8 GB 2133 MHz LPDDR3

3.3. Reflection and Suggestion

The event scheduling feature, which is available in later versions of Mesa, could be implemented for future model development to enhance efficiency. Moreover, the current data

collection process is limited to recording results only for the road named "N1" as it is already predefined. To enhance flexibility, the model should be modified to automatically retrieve and store the correct road names from the original road data, ensuring the model is applicable across different roads.

The modularity and cohesion of this code align with Hoffman's (2004) principles, ensuring necessary separation of tasks and manageable components. Each component, from BangladeshModel to individual infrastructure classes (Bridge, Sink, Vehicle), is structured with a distinct behavior, improving maintainability and scalability. The modularity allows for easy modification, for example, integrating new infrastructure types or delay mechanisms without disrupting the entire system.

However, cohesion could be slightly improved in data handling. Road names and delay tracking are currently managed separately in different parts of the code, which could be streamlined for better efficiency. Overall, the model demonstrates good modularity with well-defined agent behaviors, but it could benefit from stronger cohesion in data collection and scheduling mechanisms.

Additionally, incorporating a column indicating the condition of each bridge in the scenario data collection could have enhanced the analysis. This would have allowed for identifying the specific bridges within each category that are most critical to maintaining the functionality of the main road in Bangladesh, providing valuable insights for targeted infrastructure investment.

4. Scenario design and analysis

The simulation evaluates the impact of bridge failures on the average driving time for trucks traveling on the N1 route, Bangladesh. Eight scenarios were designed with increasing probabilities of bridge failure across four categories: A (good condition), B, C, and D (poor condition). Each scenario introduces higher failure probabilities, leading to an increasing number of broken bridges and delays. The following scenarios and results are based on the simulation of nine scenarios, each run with 10 different seeds to ensure the robustness of the implementation (see Annex B).

4.1. Simple scenario description

- **Scenario 0 (Baseline):** No bridges fail (0% probability), ensuring uninterrupted travel with no delay.
- **Scenario 1:** Only category D bridges have a 5% failure probability. Since these bridges are the least numerous (See Annex A), disruptions are minimal, causing only a slight increase in average driving time from 9.63 to 9.85 hours.
- **Scenario 2:** The failure probability for category D doubles to 10%, leading to more failures and a further increase in travel time to 10.07 hours.
- **Scenario 3:** A 5% failure probability is introduced for category C bridges while maintaining 10% for category D. As category C bridges are more numerous, disruptions intensify, resulting in an average driving time of 19.87 hours.
- **Scenario 4:** Failure probabilities increase to 10% for category C and 20% for category D, resulting in an average driving time of 24.82 hours per truck.
- **Scenario 5:** Category B bridges now have a 5% failure probability, adding another disruption layer. Failures become widespread, leading to a sharp increase in average driving time to 39.59 hours.
- **Scenario 6:** Failure probabilities rise to 10% for category B, 20% for category C, and 40% for category D. Travel is significantly disrupted, raising average driving time to 67.77 hours.
- **Scenario 7:** Category A bridges are now included with a 5% failure probability, while all other categories maintain high failure rates. Since category A bridges are the most numerous (See Annex A), even a small probability leads to considerable failures, further exacerbating delays with an average driving time of 96.33 hours.
- **Scenario 8 (Worst Case):** The highest failure probabilities are assigned: A (10%), B (20%), C (40%), and D (80%). Trucks appear unable to reach their destination, in an average driving time of zero.

Refer to Annex C for summary of results.

4.2. Scenario analysis

The simulation results highlight the impact of increasing bridge failure probabilities on both average driving time and the number of broken bridges. The trends observed across scenarios suggest an exponential relationship between failure probability and disruption severity.

Escalating Travel Disruptions

As failure probabilities increase across bridge categories, the distribution of average driving time per scenario (Figure 4.1) reveals a sharp upward trend. The shift from minimal delays in early scenarios to extreme travel times in later ones suggests that the network reaches critical failure points as more bridges become impassable. The spread of values further indicates growing variability, meaning that some simulation runs experience disproportionately higher delays, reinforcing the system's instability under stress.

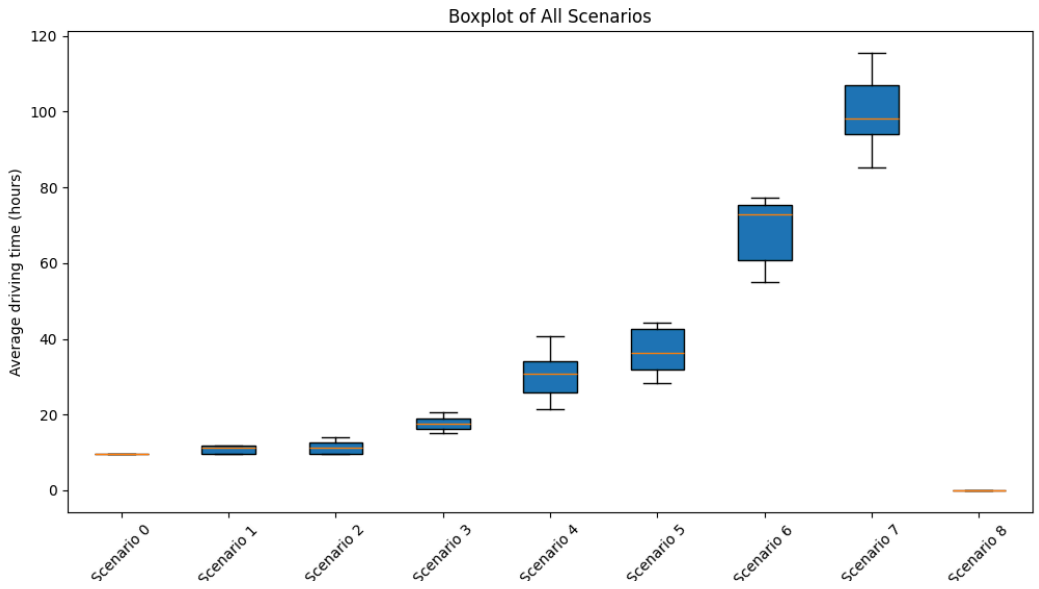

Figure 4.1: Average driving time distribution per scenario

Bridge Failures and System Breakdown

The distribution of broken bridges per scenario (Figure 4.2) follows a similar accelerating pattern. Initially, failures are limited, but as more categories are affected, the number of broken bridges surges, compounding disruptions. This exponential increase aligns with the dramatic rise in travel time seen in Figure 4.1, illustrating how infrastructure degradation leads to system-wide

inefficiencies. In the most extreme cases, bridge failures become so widespread that reaching the destination in 5 days is not possible.

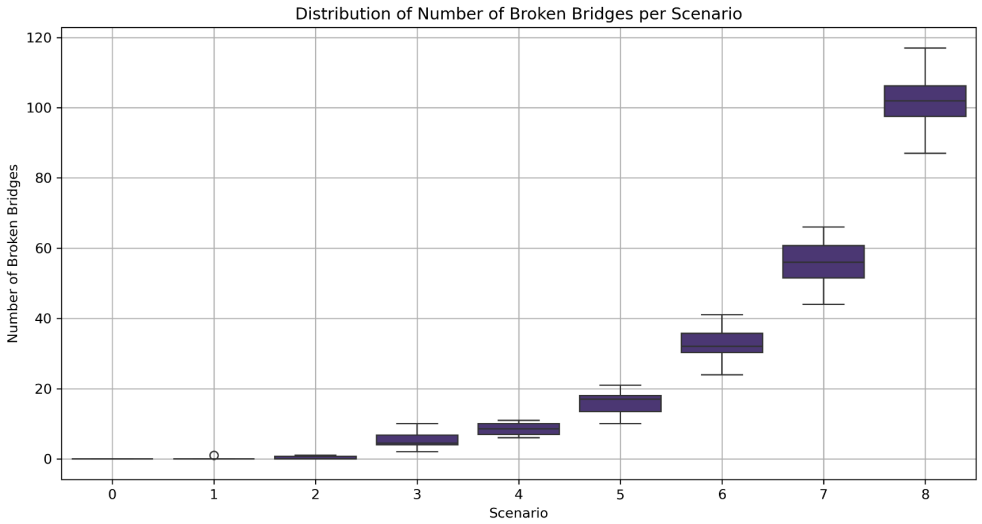

Figure 4.2: Distribution of number of broken bridges per scenario

Key Takeaways

Both figures emphasize the nonlinear nature of the disruptions. While early failures cause manageable delays, there is a tipping point where the network can no longer sustain travel, leading to collapse. This underscores the importance of maintaining infrastructure, as even small increases in failure probabilities can have disproportionate effects on overall transport efficiency.

5. Bonus exercises

To identify the top five bridges in the most critical condition, a bridge prioritization calculation ranks them based on their condition and length. The process is as follows:

- Assign condition scores: Condition A = 1, B = 2, C = 3, D = 4 (higher means worse condition).
- Normalize bridge length: Scale the length between 1 and 4, ensuring that longer bridges receive higher scores.
- Weighted scoring system: Assign a greater importance to condition (0.7) compared to length (0.3).
- Finalize the priority score, sort bridges (descending). Select the top five bridges for investment.

The results of the bridge prioritization indicate that bridges in poor condition (categories D and C) are ranked higher. Among bridges with the same condition rating, longer bridges receive higher priority scores, as their failure would significantly impact traffic delays. The top five bridges include:

1. Morichia (N1_LRP394a)
2. Shapur Steel Beam and RCC Slab (N1_LRP396b)
3. Lemua Bridge (N1_LRP155a)
4. Sarkar Bari Culvert (N1_LRP046a)
5. Shammoti Para (N1_LRP382c)

These bridges require immediate attention due to their condition and structural significance. On top of that, while handling mechanisms should depend on each bridge's condition and needs, maintenance should be applied to all bridges, especially the most frequently driven on bridges—which could be data useful to collect.

References

Hofmann, Marko. (2004). Criteria for Decomposing Systems Into Components in Modeling and Simulation: Lessons Learned with Military Simulations. Simulation. 80. 357-365. 10.1177/0037549704049876.

Annex

A. Bridge condition and count

Table 1. Bridge condition and count

Condition	Bridge count
A	421
B	113
C	88
D	6

B. Simulation results

Average_driving_time, Total_waiting_time, and Average_waiting_time are in minutes. Upon running '*model_run.py*', a column named Broken_bridges will be filled with a list of broken bridges. However, said column has been excluded from these tables to avoid an exhaustive annex.

1. Scenario 0: {'A': 0.0, 'B': 0.0, 'C': 0.0, 'D': 0.0}

Table 2. Scenario 0

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	0	411917	578.0	0	0.0
N1	0	371785	578.0	0	0.0
N1	0	831836	578.0	0	0.0
N1	0	645051	578.0	0	0.0
N1	0	483421	578.0	0	0.0
N1	0	395845	578.0	0	0.0
N1	0	770680	578.0	0	0.0
N1	0	718908	578.0	0	0.0
N1	0	397034	578.0	0	0.0
N1	0	208709	578.0	0	0.0

2. Scenario 1: {'A': 0.0, 'B': 0.0, 'C': 0.0, 'D': 0.05}

Table 3. Scenario 1

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	1	693113	708.0115473	172549	132.8321786
N1	1	183602	707.077812	171445	132.0839753
N1	1	994883	707.3938462	171749	132.1146154
N1	1	637885	645.1596639	89245	68.17799847
N1	1	111868	708.6140216	173923	133.9930663
N1	1	108757	578	0	0
N1	1	834793	578	0	0
N1	1	961574	707.8527371	172399	132.921357
N1	1	486464	578	0	0
N1	1	424301	578	0	0

3. Scenario 2: {'A': 0.0, 'B': 0.0, 'C': 0.0, 'D': 0.10}

Table 4. Scenario 2

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	2	826156	645.8237986	90020	68.66514111
N1	2	758017	578	0	0
N1	2	346020	708.1960031	172711	132.7524981
N1	2	163507	578	0	0
N1	2	297588	707.5330769	170348	131.0369231
N1	2	938322	578	0	0
N1	2	477272	837.0990566	339049	266.547956
N1	2	751339	774.3496503	257361	199.969697
N1	2	697883	578	0	0
N1	2	905706	836.2492138	340798	267.922956

4. Scenario 3: {'A': 0.0, 'B': 0.0, 'C': 0.05, 'D': 0.10}

Table 5. Scenario 3

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	3	554600	942.0031898	479551	382.4170654
N1	3	512436	905.4405705	428566	339.59271
N1	3	661155	965.1947115	511077	409.5168269
N1	3	771707	973.4743178	517327	415.1902087
N1	3	941123	1038.761943	605246	490.0777328
N1	3	596349	1132.294069	721313	594.1622735
N1	3	345865	1231.613579	842453	706.1634535
N1	3	277341	1070.971382	654549	535.1995094
N1	3	224558	1228.675856	852695	712.3600668
N1	3	869327	1137.929043	744682	614.4240924

5. Scenario 4: {'A': 0.0, 'B': 0.0, 'C': 0.10, 'D': 0.20}

Table 6. Scenario 4

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	4	236002	2445.006303	2191320	2301.806723
N1	4	278219	1289.235443	918862	775.4109705
N1	4	282675	1421.62825	1086606	941.5996534
N1	4	739895	1814.491163	1518039	1412.129302
N1	4	738677	1879.230914	1618850	1525.777568
N1	4	433903	1953.83874	1658753	1582.77958
N1	4	295797	2074.183415	1815519	1771.238049
N1	4	549698	2121.479331	1877558	1847.990157
N1	4	513386	1489.416155	1166717	1024.334504
N1	4	949718	1756.618698	1464991	1342.796517

6. Scenario 5: {'A': 0.0, 'B': 0.05, 'C': 0.10, 'D': 0.20}

Table 7. Scenario 5

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time

N1	5	528267	2013.973025	1761469	1696.983622
N1	5	512605	2662.417219	2424497	2676.045254
N1	5	136065	1873.544685	1606225	1511.030103
N1	5	338296	2242.400605	2016927	2035.244198
N1	5	737695	1699.701181	1416326	1286.399637
N1	5	464005	2118.288102	1876425	1845.058997
N1	5	349368	2418.728796	2163963	2265.929843
N1	5	833734	1822.375582	1578277	1470.901212
N1	5	466115	2605.718172	2364678	2573.099021
N1	5	892961	2639.351322	2407386	2651.306167

7. Scenario 6: {'A': 0.0, 'B': 0.10, 'C': 0.20, 'D': 0.40}

Table 8. Scenario 6

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	6	980180	3710.119829	3344535	4771.091298
N1	6	846891	4376.784698	3730193	6637.354093
N1	6	637172	4553.492424	3851153	7293.850379
N1	6	417483	3583.048476	3203464	4436.930748
N1	6	168557	4398.661947	3786075	6701.017699
N1	6	243513	4585.564547	3864926	7446.870906
N1	6	920537	4636.373047	3950827	7716.458984
N1	6	450420	3302.348329	2967896	3814.77635
N1	6	175022	3617.586835	3236052	4532.285714
N1	6	129965	4372.573684	3771822	6617.231579

8. Scenario 7: {'A': 0.05, 'B': 0.10, 'C': 0.20, 'D': 0.40}

Table 9. Scenario 7

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	7	329655	6215.328283	4575435	23108.25758
N1	7	257555	6929.28	4741767	189670.68

N1	7	882811	5655.220395	4409028	14503.38158
N1	7	383704	5888.94636	4435953	16995.98851
N1	7	277037	6685.377551	4616843	47110.64286
N1	7	745562	5630.61859	4362667	13982.90705
N1	7	127841	5112.794621	4142228	10127.69682
N1	7	626202	5880.444015	4437899	17134.74517
N1	7	438451	6487.116788	4579508	33427.06569
N1	7	549855	5582.900929	4382555	13568.28173

9. Scenario 8: {'A': 0.10, 'B': 0.20, 'C': 0.40, 'D': 0.80}

Table 10. Scenario 8

Road	Scenario	Seed	Average_driving_time	Total_waiting_time	Average_waiting_time
N1	8	553782	0	4745074	0
N1	8	337912	0	4847917	0
N1	8	304999	0	4867121	0
N1	8	251520	0	4885570	0
N1	8	318205	0	4875667	0
N1	8	873825	0	4888428	0
N1	8	208218	0	4849037	0
N1	8	573559	0	4818808	0
N1	8	917164	0	4874810	0
N1	8	295980	0	4863911	0

C. Simulation Summary

Summary of average driving time per scenario.

Table 11. Simulation Summary

Scenario	Average driving time (hours)
0	9.63
1	9.85

2	10.07
3	19.87
4	24.82
5	39.59
6	67.76
7	96.33
8	0 (no truck reached the sink)

Acknowledgement

The use of AI

Use of AI in Code Development: In this assignment, AI tools such as GitHub Copilot and ChatGPT were used to improve code quality and generate ideas for implementation. These tools assist in structuring code, suggesting optimizations, and debugging issues. However, AI's role was strictly supportive—we conducted final decisions, implementations, and refinements to ensure accuracy and adherence to project requirements.

Use of AI in Writing and Editing: ChatGPT and Grammarly were used for text improvement, helping refine clarity, coherence, and grammatical correctness. It assisted in brainstorming alternative phrasings and ensuring correct writing standards. However, all AI-generated suggestions were critically reviewed, modified when necessary, and integrated into our work only after careful validation. This approach ensured that AI served as a tool rather than a replacement for our original analysis and insights.

Contribution of each member

Member	Contribution
Rachel Delvin Sutiono	- Component building and modeling - Report
Celia Martínez Sillero	- Component building and modeling - Report
Daniela Ríos Mora	- Component building and modeling - Report
Thunchanok Phutthaphaiboon	- Demo file creation - Report
Yao Wang	- Demo file creation - Report