CSE6730 Modeling and Simulation

Dynamic Traffic Network Simulation and Impact of Pluvial Flash Flood

Group 9

Atticus Rex
Garyoung Lee
Li-Yen Yang
Yifan Zhao
Flavia Kung'u

Submitted: December 5th, 2023

Github Repository: Click Here

Video: Click Here

Table of Contents

Abstract

In contemporary transportation systems, the management of traffic flow under heavy load and unpredictable events such as flash floods are a critical challenge. The conventional static traffic assignment, where routes are fixed at the onset, is insufficient in addressing real-time variations in trip volumes, accidents, work zones, and weather conditions. Dynamic Traffic Assignment (DTA) is a pragmatic solution, adapting routing strategies continuously to evolving traffic conditions. Existing navigation services, while adept at adapting to incidents like accidents and road closures, often neglect unreported events like flash floods, posing a significant threat to both drivers and system integrity.

Unlike accidents or work zones, flash floods exhibit distinct characteristics: they spread spatially and temporally, impede rerouting options when vehicles are submerged, cause physical damage, confound drivers with safe detours, and significantly reduce network capacity. Despite the gravity of this issue, an efficient and safe rerouting strategy tailored for flash floods remains elusive in existing research.

Addressing this gap, our research aims to develop a bespoke DTA model. This model is built to respond to flash floods, ensuring the shortest and safest travel time for drivers by rerouting vehicles dynamically and optimizing paths to avoid flooded areas. Our approach will help contribute to a DTA algorithm that integrates real-time data on flash floods, accidents, and road closures.

By simplifying the traffic network setting, we will replicate diverse flooding intensities, enabling comprehensive evaluation of our dynamic traffic assignment strategy. By systematically analyzing these simulations, our research aims to establish a robust and adaptive framework for rerouting under flash flood conditions, thereby enhancing the resilience and safety of modern transportation networks.

1. Project description

Urban traffic systems are inherently complex, particularly during peak hours when excessive demand and limited capacity lead to congestion. This complexity is compounded by adverse weather conditions such as heavy rain, which can lead to flash floods and further challenges traffic management. For individual vehicles, finding the shortest path in a network might be straightforward. However, optimizing routes for hundreds of thousands of trips in an urban network requires a more sophisticated approach. As roads reach capacity, the speed of travel decreases, leading to congestion, particularly during rush hours. Rain exacerbates these issues, reducing road speeds and, in severe cases, causing flash floods that inundate roads and necessitate traffic rerouting due to the deadlock.

In response to these challenges, this project aims to build a Dynamic Traffic Assignment (DTA) simulation to optimize traffic routes in urban road networks, focusing on the impact of road capacity combined with traffic flows and flash floods on traffic assignment. The model transforms an urban road network into a digital framework of nodes and links, referred to as Network, with each node representing intersections or points of interest, and links depicting the roads connecting these nodes. Attributes such as link length, speed limit, and capacity are assigned to each link. The traffic demand is described as an Origin and Destination (OD) matrix with origin nodes and destination nodes denoted as row and column headers. The values of each cell in the OD matrix represent the number of trips from and arriving at corresponding OD pair. The traffic assignment phase employs dynamic methods to determine optimal routes based on criteria of shortest travel time and network efficiency, using approaches of All-or-Nothing (AoN), and System Optimal (SO).

Given the time constraints, the scope of this project is confined to a simplified N by N arbitrary transportation network initially with free-flow travel times and a randomly but realistically generated OD matrix. A unique aspect of this model is its capability to adapt to varying flood conditions, adjusting the capacity of links affected by floods to reroute traffic away from inundated areas. This model provides insights into managing urban traffic efficiently under both normal and adverse flash flood conditions.

2. Literature review

The critical impact of flash floods on urban traffic routing is well-documented, with studies emphasizing the need for dynamic traffic models that can adapt to such extreme conditions. Choo et al. (2020) highlight the significant disruption caused to traffic flow due to urban floods, establishing a direct relationship between rainfall depth and vehicle speed reduction. This correlation is pivotal in understanding the necessity for traffic rerouting during flash floods (Choo, Kang, & Kim, 2020). Further emphasizing the urgency, Pregnolato et al. (2017) and Diakakis et al. (2020) discuss the escalating impact of climate change on pluvial flooding and its severe implications for transportation infrastructure, underlining the importance of incorporating these factors into traffic models (Pregnolato et al., 2017; Diakakis et al., 2020).

Flash flood modeling typically employs a combination of two sub-models. The first integrates precipitation data with drainage and topography models to identify areas prone to flooding and assess changes in road networks during flash flood scenarios. The second model focuses on traffic routing, adapting to the altered road networks determined by the first model. This interaction between flash floods and traffic, especially under varying climate conditions, is explored through diverse drainage and routing models in contemporary studies (Blöschl, Reszler, & Komma, 2008; Jamali, Bach, Cunningham, & Deletic, 2019; Kouroshnejad et al., 2023). In this project, we focus on the development of the traffic routing model, employing dynamic traffic assignment techniques. The flood-affected areas and schedules are arbitrarily determined, simulating the progression from low to high inundation over time. According to Sheffi (1985) and Peeta and Ziliaskopoulos (2001), the dynamic traffic assignment effectively provides flexibility and responsiveness required to manage traffic under rapidly changing network conditions, such as those caused by flash floods.

Incorporating the effects of precipitation and different inundation levels into the traffic model is crucial. As demonstrated by Li et al. (2018) and Rebally et al. (2021), these factors can be effectively modeled through variations in road capacity. Our approach categorizes flood severity into three distinct levels—wet surface, moderate flood, and high flood—and correspondingly adjusts link capacities to reflect these conditions (Li et al., 2018; Rebally et al., 2021).

3. Conceptual model

The traffic assignment should be based on two basic elements: road network and traffic demand. Therefore, our DTA model takes into two variables: Network and an OD matrix.

The Network is a digital framework of nodes and links, with each *node* representing intersections or points of interest, and *link* depicting the roads connecting these nodes. An example grid network of 2 by 2 is shown in Figure 3.2. The nodes and links have attributes including length, speed limit, capacity, initial travel time (free-flow travel time at speed limit), and the responsive travel time based on the current flow of the link. The responsive travel time is determined by the BPR function ((U.S. Bureau of Public Roads, 1964) as following:

$$t = t_0[1 + \alpha \left(\frac{V}{C}\right)^{\beta}] \tag{3.1}$$

where,

t is the responsive travel time

t_0 is the initial free flow travel time = link length / speed limit

V is the flow (volume) on the link (number of vehicles)

C is the road capacity

β, α are parameters to be calibrated (4 and 0.15 are typically used)

The illustrative relationship between responsive travel time and flow described by the BPR function is demonstrated in Figure 3.1. As the flow increases, the travel time to pass the link will be increased, reflecting the real-world scenario in which heavy traffic leads to lower traffic speed and therefore longer travel time.

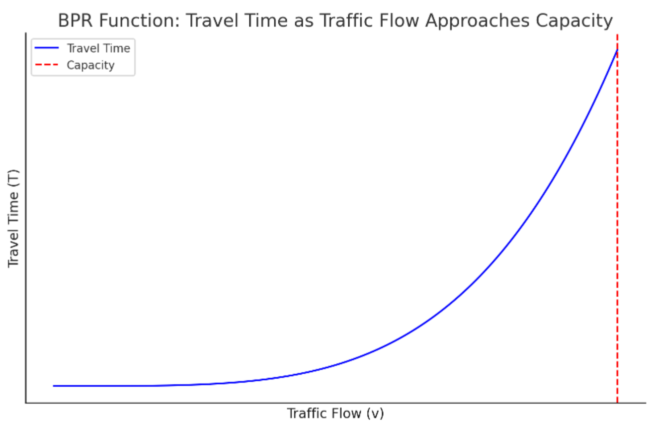


Figure 3.1: Travel time vs. flow subjective to BPR function

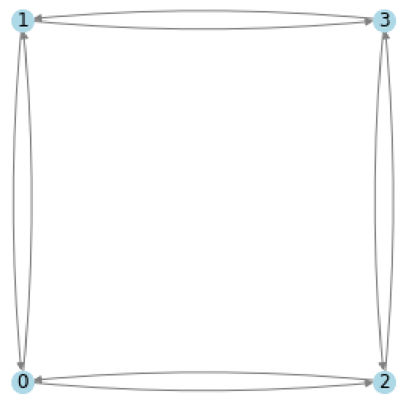


Figure 3.2: An example of 2X2 network

The OD matrix contains the information about the distributed trip across the network, where all nodes are both the origin and destination. The values of each cell in the OD matrix represent the number of trips from the origin and destination. The shaded area in Table 3.1 gives an example of OD matrix for the network in Figure 3.2, which describes there are ten trips from node 1 to node 2 and six trips from node 3 to node 2.

Table 3.1: OD matrix for the 2X2 network

	1	**2**	**3**	**4**
1	0	10	20	10
2	5	0	7	9
3	8	6	0	4
4	9	11	15	0

There is an imaginary central traffic management center to assign the trips in the OD matrix with certain routes in order to optimize the travel time for each trip, such as Google Map in reality. Our DTA model works as such a management center which takes into the Network and OD matrix as input and produces a set of paths considering the shortest time for trip demand of each OD pair and the network constraints. Here we assume that the trips from the same OD will be assigned to the same path. The traffic assignment will be subjective to the link capacity, so the model would make sure the assigned flow would not exceed the capacity of any link.

The DTA model will: (i) route trips based on the current network states and (ii) update the network flows and corresponding responsive travel times based on the assigned routes. Initially, we must initialize the flows on all links using the given OD matrix and update the responsive travel time. Subsequently, the optimal route will be assigned for all OD pairs. When we halt at this stage by providing route information for the trip independently of time, we refer to it as static traffic assignment. However, to account for dynamic changes, we should update the current flows and the corresponding shortest path at desired time intervals. The DTA model recalculates the optimal paths through iterations. As the system is constrained by capacity, the model might provide different routes for the same OD in each iteration. Although the updated paths may or may not maintain the total system travel time lower than the initial static scenario, the traffic assignment system can respond to changes promptly. Overall, the DTA model dynamically routes traffic according to the current flow on the network to maintain being optimized.

The impact of flash flood is incorporated into the model with the reduced road capacity. The dynamic model would react with rain or flash floods with dynamic routing based on the flood-induced capacity drop. Therefore, the traffic would avoid the heavily inundated area and flow less into the inundated area to maintain an overall optimal travel time.

The impacts are categorized into three levels, which correspond to (i) wet surface, (ii) moderate flood/inundation and (iii) heavy flood/inundation. When it starts to rain, the road becomes wet, and all road capacities are slightly reduced as the drivers tend to slow down even in a free flow, making the speed limit unavailing. As it rains and certain areas get inundated moderately, e.g., the shoulder lane is flooded, the road capacities will be further reduced for these links only. And when heavy flood happens somewhere in a link, the link would be closed. The flash flood schedule will be created in order to be more realistic as inundation does not appear abruptly but grow as rain falls.

Figure 3.3: An example of moderate flood

Figure 3.4: An example of heavy flood and corresponding road closure

4. Simulation

4.1 Overview of our software implementation

Our simulation model is implemented using Python. The primary Python libraries we used in the software are: NetworkX, NumPy, Matplotlib, and SciPy. In the GitHub repository we provided, the final version of our code implementation is in the `/FinalDeliverable` directory. Functions for the simulation are written in *`functions.py`*. Functions for visualizing the network are written in *`visualize.py`*. The results of the visualizations are stored in `/FinalDeliverable/visualizations`. To run the simulation, execute the *`main.py`* file.

Below are the commands to install and run the simulation:

```
git clone https://github.gatech.edu/glee388/2023Fall-CSE6730-Group9

cd FinalDeliverable

python main.py
```

To implement a simulator for the traffic flow, we defined a few functions. These functions can be mainly classified in the following categories.

4.1.1 Network Definition:

These functions are used to define a traffic network. Our simulation model uses grid networks to represent traffic situations. They are implemented using **grid_2d_graph** with a **directed** graph class in NetworkX. Each 2d graph has N^2 nodes with bi-directional edges connecting the neighbor nodes. Recall Figure 3.2

showing an example grid network with four nodes. In the scenario of a traffic simulation, each node represents a location of interest, and each edge can be seen as the street connecting one location to the other.

Within each edge, we assign several different attributes to describe the edge connecting a pair of nodes. The attributes include length, flow, speed limit, capacity, travel time and responsive travel time. The description of each attribute is described below:

Length - the length of the street/edge.

Flow - the current traffic flow (number of vehicles) on the street/edge.

Speed limit - the maximum speed a vehicle can travel on the street/edge.

Capacity - the maximum flow allowed on the street/edge at a time.

Travel Time – Free-flow travel time = Length / Speed limit

Responsive Travel Time - Time to travel from one end to the other end of the street/edge given the current traffic situation.

Related function: `create_grid_network` (see `functions.py` for details)

4.1.2 OD matrix generator:

These functions are used to generate the OD matrix. For a network with N^2 nodes, an OD matrix will have a shape of N^2 x N^2. The value of O_iD_j represents the number of trips from an origin node i to a destination node j. The OD matrix can be generated using different distributions that describe the demand of trips. We implemented two different ways of generating the OD matrix. The first way is to generate an OD matrix to randomly assign the number of trips between an origin and destination using a Poisson Distribution. The second way is to generate an OD matrix to simulate the trip distribution more realistically - the assumption is based on the central business district (CBD) model, where most people tend to depart from the boundary and travel to the center of the city.

Related functions:

`generate_random_od_matrix`

`generate_CBD_od_matrix`

(see `functions.py` for details)

4.1.3 Traffic Assignment:

The dynamic traffic assignment iterates through a user-defined number of iterations to assign the traffic flow dynamically. In each iteration, the assignment is processed as follows: (1) Calculate the flow on each edge; (2) Find k-shortest path for each trip based on the responsive travel time; (3) For all the trips with the same origin and destination, distribute the flows across these paths that takes the current available capacity into considerations; (4) Update the responsive travel time on each edge

Related functions (see functions.py for details):

```
dynamic_traffic_assignment
initialize_traffic_assignment
Update_traffic_assignment
```

4.1.4 Flood Simulation:

This function simulate the flash flood conditions by changing the capacity of the edges that are affected by the flood situations. The situation of the flood can be simulated in three different settings: (1) light rain: reduce the capacity of the impacted edges to 95% of the original capacity (2) moderate flood: reduce the capacity of the impacted edges to 50% of the original capacity (3) severe flood: reduce the capacity of the impacted edges to 0% of the original capacity.

Throughout the simulation, we can update the flood situation and impacted edges iteratively with a schedule. For example, we can simulate the flood starting from light rain to moderate flood and to severe flood. We can also define the edges/streets that are impacted and change it spatiotemporally in each iteration.

Related functions (see functions.py for details):

```
update_network_for_flood
```

4.1.5 Visualization

At each time interval of the dynamic traffic process, we visualized the travel time, traffic flow, and flood intensity on each edge. We also visualized the shortest path from one origin to one destination node based on the traffic assignment.

Related functions (see visualize.py for details):

`Visualize_traveltime`

`Visualize_flow`

`Visualize_shortest_path`

`visualize_flood`

4.1.6 Other Calculation Functions and Update Functions:

Lastly, there are a few helper functions for calculating network related properties or updating the travel time and flow on the network graph.

Related functions (see functions.py for details):
`get_total_travel_time`

`update_flow_and_TT`

4.2 Overview of the simulation process:

In our `main.py` script, we demonstrated how we can simulate the system:

Step 1: Define network parameters and flood parameters

```
# System Parameter
seed = 42 # random seed
dynamic_iterations = 12

# Network Parameter
N = 6 # Size of grid (N X N)
L = 0.2 # Length of link (km)
V_f = 50 # Speed limit (km/h)
veh_len = 0.005 # Length of vehicle (km)
lane_per_dir = 1 # Lanes per direction
reaction_time = 1.2 # Reaction time (s)
OD = 'CBD'  # Select between 'random' and 'CBD'
density_level = 0.3 # Total number of vehicles are controlled by density_level * total sum of capacities in the network

# Flood-related Parameter
flooding = True
flood_impacts = {
'light_rain': 0.95,   # 98% of original capacity
'moderate_flood': 0.50,  # 50% of original capacity
'severe_flood': 0.0000000001,    # 0% of original capacity (impassable)
}

flood_schedule = { # Modify flood schedule. key: the iteration number that certain level of flood will be applied
    4: {'level': 'light_rain', 'edges': []},  # Apply to all edges
    5: {'level': 'moderate_flood', 'edges': [((0,0), (0,1)), ((1,0), (2,0)), ((2,1), (2,2)), ((2,2), (3,2))]},
    6: {'level': 'moderate_flood', 'edges': [((4,1), (4,0)), ((4,0), (4,1)), ((4,1), (4,2)), ((4,2), (4,3))]},
    10: {'level': 'severe_flood', 'edges': [((2,1), (2,2)), ((2,2), (3,2)), ((0,2), (1,2)), ((0,0), (1,0)),((1,0), (0,0))]},
    11: {'level': 'severe_flood', 'edges': [ ((0,2), (1,2)), ((0,0), (1,0)),((1,0), (0,0))]},
}
```

Step 2: Initialize the network and visualize the initial network

```
print("Initializing Network...")
# Creating the graph
G, pos, labels = create_grid_network(N, L, V_f, veh_len, lane_per_dir, reaction_time)
# Generating OD Matrix
if OD =='CBD':
    od_matrix = generate_CBD_od_matrix(G, N, density_level, seed, CBD_weight=5)
else:
    od_matrix = generate_random_od_matrix(G, N, density_level)
# Initialize Traffic assignment
shortest_paths = initialize_traffic_assignment(G, od_matrix, N, k=3)
# Updating flow and travel time
update_flow_and_TT(G, shortest_paths, od_matrix, N)
# Visualizing shortest paths
total_time = get_total_travel_time(G, shortest_paths, od_matrix, N)
# Visualizing travel time and flow
visualize_traveltime(G, pos, labels, total_time, filename='visualizations/TT_initial.png')
visualize_flow(G, pos, labels, filename = 'visualizations/Flow_initial.png')
visualize_shortest_path(G, pos, shortest_paths, (0,0), (5,5), labels, filename = 'visualizations/shortest_path_initial.png')
```

Step 3: Run iterations of Dynamic Traffic Assignment and Visualize the Results

```
# Dynamically Updating traffic assignment
for iteration in range(dynamic_iterations):
    print("Iteration %d" % (iteration + 1))
    if flooding:
        update_network_for_flood(G, iteration, flood_schedule,flood_impacts, {(u, v): d['capacity'] for u, v, d in G.edges(data=True)})
        shortest_paths = update_traffic_assignment(G, od_matrix, N, k=3)
        visualize_flood(G, pos, labels, flood_schedule, flood_impacts, iteration, filename='visualizations/Flood_Capacity_%d.png' % (iteration))
    else:
        shortest_paths = update_traffic_assignment(G, od_matrix, N, k=3)
    # Calculating total time
    total_time = get_total_travel_time(G, shortest_paths, od_matrix, N)
    # Visualizing travel time
    visualize_traveltime(G, pos, labels, total_time, filename = 'visualizations/TT_%d.png' % (iteration))
    visualize_flow(G, pos, labels, filename = 'visualizations/Flow_%d.png' % (iteration))
    # Visualizing shortest paths
    visualize_shortest_path(G, pos, shortest_paths, (0,0), (5,5), labels, filename = 'visualizations/shortest_path_%d.png' % (iteration))
```

4.3 Verification:

After implementing the simulator, we verified if our model is implemented correctly by visualizing the flows, travel time, and shortest paths returned in each iteration and checking whether the results match the parameter settings from the conceptual model.

5. Experimental results and validation

Modeling inputs:

- Network: The network is a 6x6 network. The speed limit and capacity of roads and nodes in the inner center nodes is lower than the outer nodes to represent a CBD pattern.
- OD matrix: The OD matrix is randomly generated but presents a CBD pattern in which more trips are originated from boundary nodes and going to the CBD nodes.
- Flood impact:
  Light rain: 95% of original capacity
  moderate flood: 50% of original capacity
  severe flood: 0% of original capacity (impassable)
- Flood schedule:

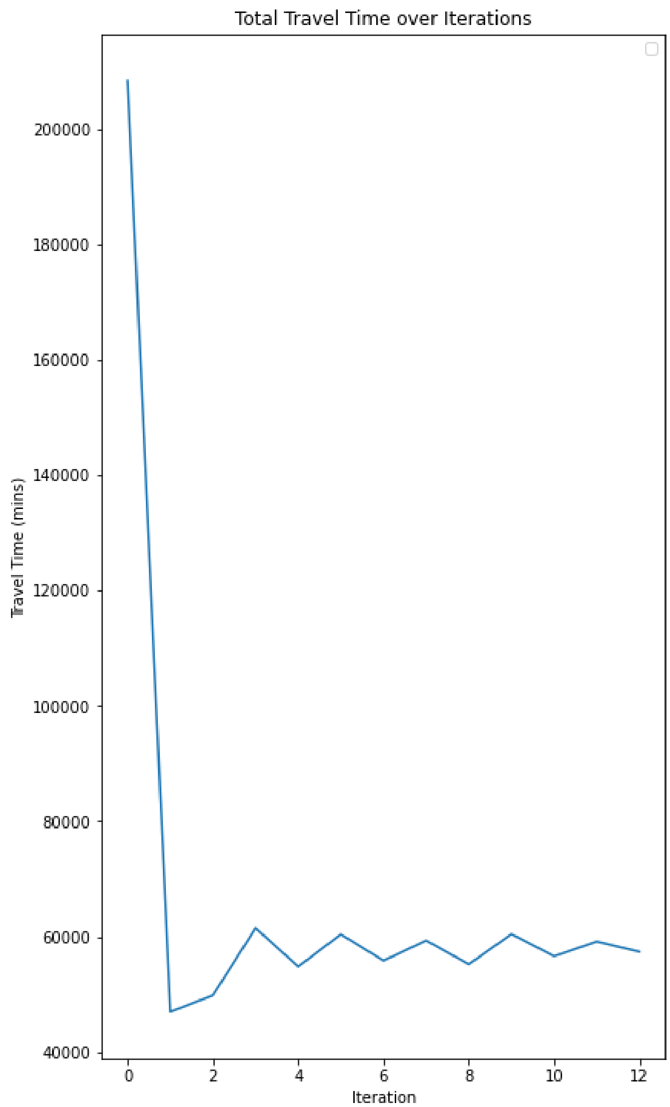
Time	Event	Impact area
4	Light rain	The whole network
5	Moderate flood	[(0,0),(0,1)] [(1,0),(2,0)] [(2,1),(2,2)] [(2,2),(3,2)]
6	Moderate flood	[(4,1),(4,0)] [(4,0),(4,1)] [(4,1),(4,2)] [(4,2),(4,3)]
10	Severe flood	[(2,1),(2,2)] [(2,2),(3,2)] [(0,2),(1,2)] [(0,0),(1,0)] [(1,0),(0,0)]
11	Severe flood	[(0,2),(1,2)] [(0,0),(1,0)] [(1,0),(0,0)]

5.1 Dynamic Traffic Assignment Simulation without Flood

After running the DTA models for 13 iterations (including 1 initial assignment (iteration 0) and 12 dynamic assignment iterations), the first significant result we obtained is that the travel time is largely reduced from the initial condition and maintained with iterations as clearly demonstrated in the Figure 5.1A, which plots the total travel time over iterations. This huge drop corresponds to the change in traffic assignment algorithm. In the initial iteration, routing is based on free flow speed and capacity constraint, but the actual travel time would be much lower than estimated when conduct routing because the flow's impact on link travel time described in BPR function.

The travel time of each link is depicted in Figure 5.1B for the initial condition and first 2 iterations from top to bottom. Comparing the initial travel time diagram with the latter two, the overall traffic is bad at the initial condition with red and orange links spreading out the network. For the later on iterations shown in Figure 5.1B, the travel time of each link reflect a CBD pattern with higher travel time at the inner links and lower travel time at the outer links. The travel time diagrams for the dynamic assignment iterations present pattern that closer to the actual attributes to the road Network and OD matrix, which demonstrate a more successful traffic assignment than the initial one without considering the traffic flow on the road.

As shown in Figure 5.1C, this pattern is kept throughout iterations even though locations of red links change across iterations. This observation reflects the function of the model to dynamically assign traffic according to the current network situation to maintain an optimal travel time.



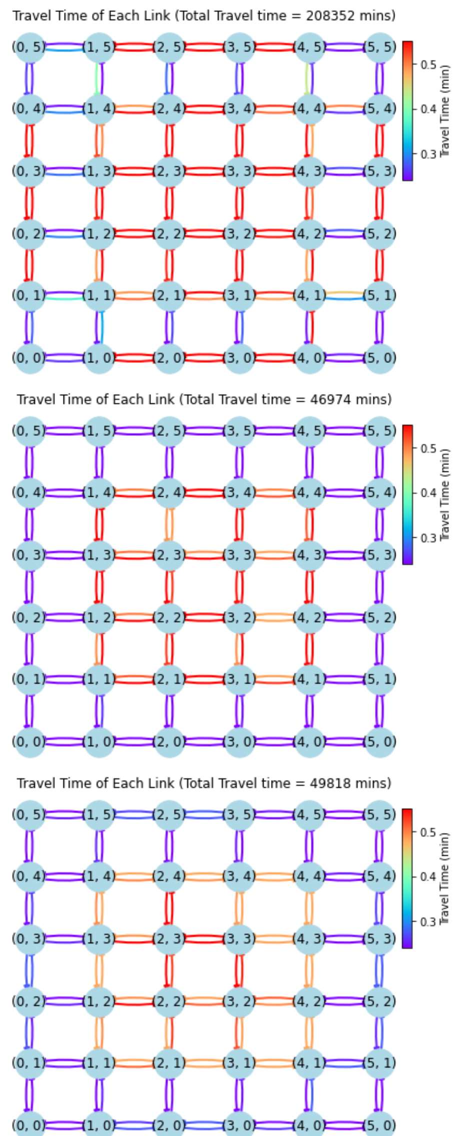


Figure 5.1A (left): The total travel time change over iterations
Figure 5.1B (right): The responsive travel time of each link for the initial condition and first 2 dynamic assignment iterations (from top to bottom)

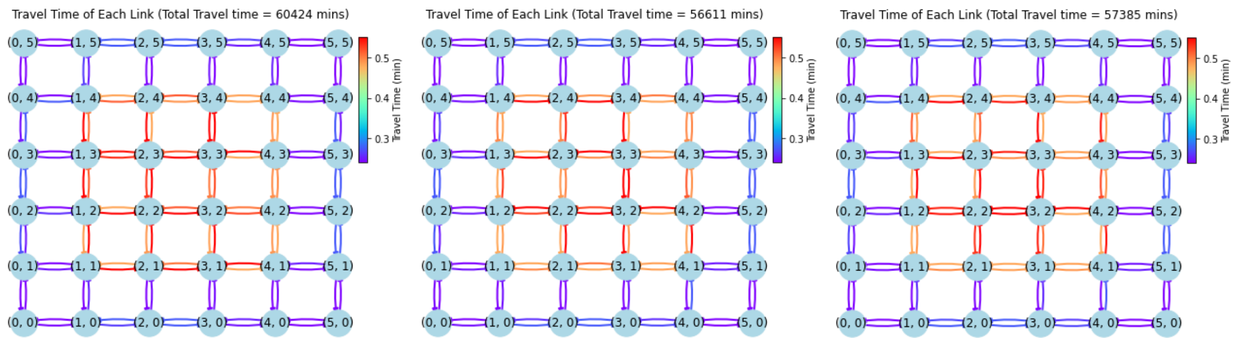


Figure 5.1 C: The responsive travel time of each link for the 8th, 10th and 12th dynamic assignment iterations (from left to right)

Figure 5.1D shows the shortest path assignment to trip form (0,0) to (5,5) at different iterations with travel time for a single trip displayed. The travel time of 6.87 minutes is largely reduced at the first 3 iterations and kept around 2.6 minutes. The shortest paths are different from iteration to iteration. These observations further justify that the DTA model could successfully conduct optimal routing results based on the current and changing road situation.

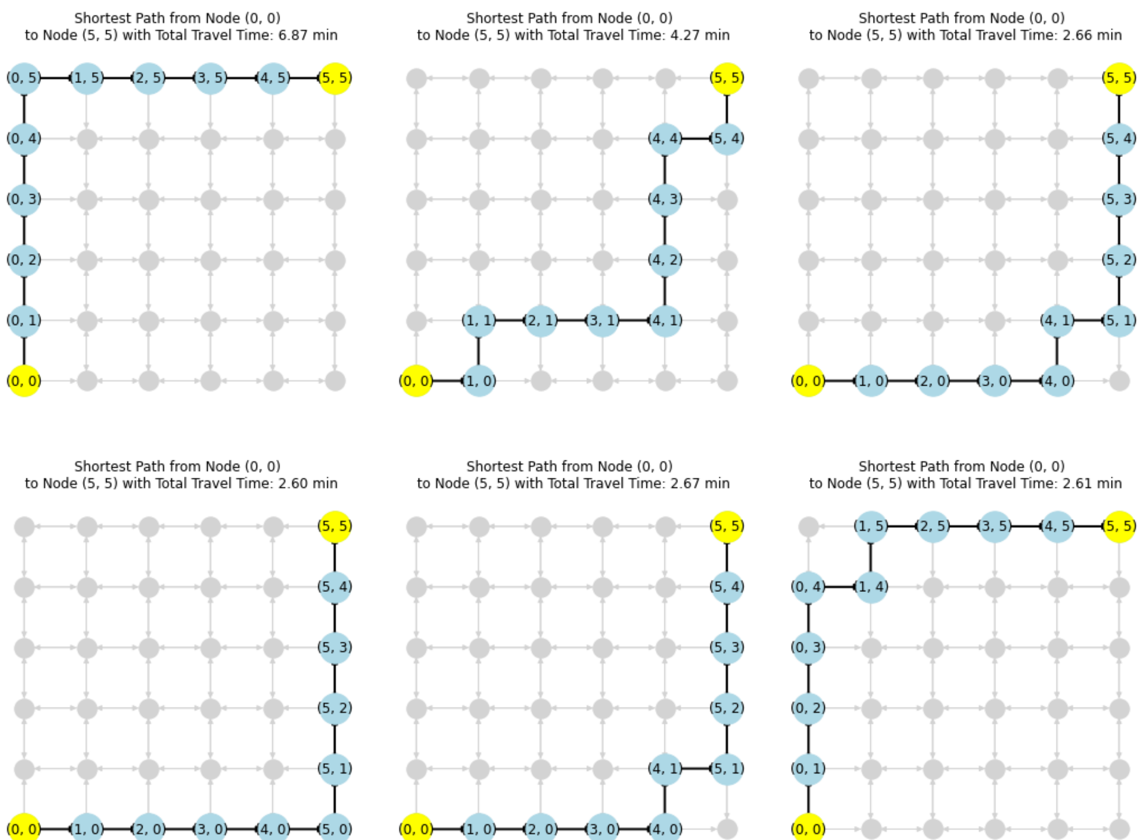


Figure 5.1 D: The shortest paths assigned for trip (0,0) to (5,5) with travel time for a single trip at the 0th, 1st, 2nd, (first), 8th, 10th, 12th (second row) dynamic assignment iterations (from left to right)

5.2 Dynamic Traffic Assignment Simulation with Flood

We also ran the DTA models for 12 iterations with scheduled flood conditions, starting from light rain to severe flood. Figure 5.2A shows the total travel time of all vehicles throughout the iterations. Figure 5.2B shows the visualization of the network. From left to right, we show the capacity, travel time, and traffic flow of the network. From top to bottom, we show the visualization of different iterations through time. The first row shows the initial conditions of the network, and the second to the last rows show the conditions of the network from the first iteration to the last iteration. From these plots, we can observe that even though the capacity of several impacted roads has significantly decreased as a result of the flash flood situation, the total travel time still remains reasonable. Additionally, when severe flood happens at the 10 and 11 iterations, the DTA model successfully assigned traffic so that the severely flooded are avoided. From these observations, we can conclude that the DTA model could react with flood situation and help optimize the travel time, as it does not increase significantly compared to the simulation without flood.

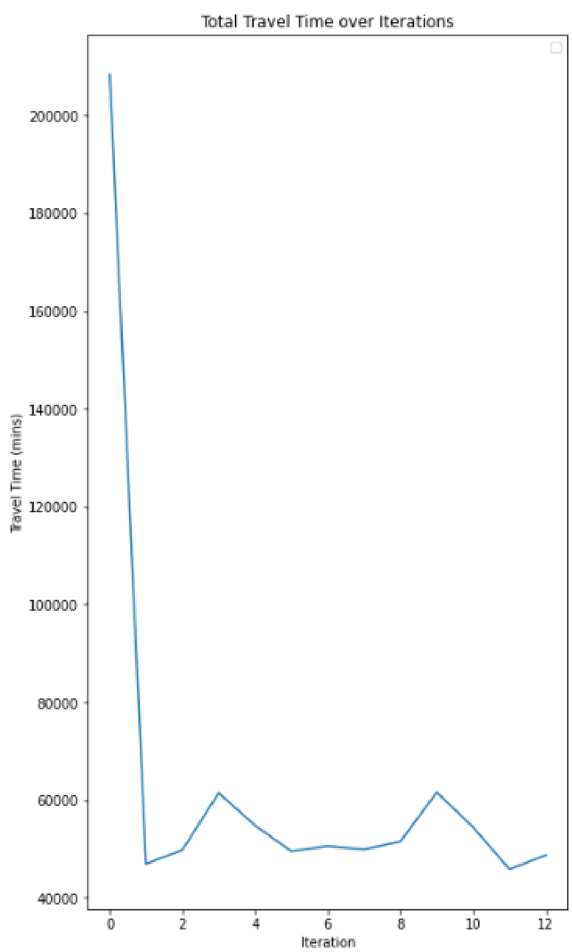


Figure 5.2B Total Travel Time throughout the Iterations

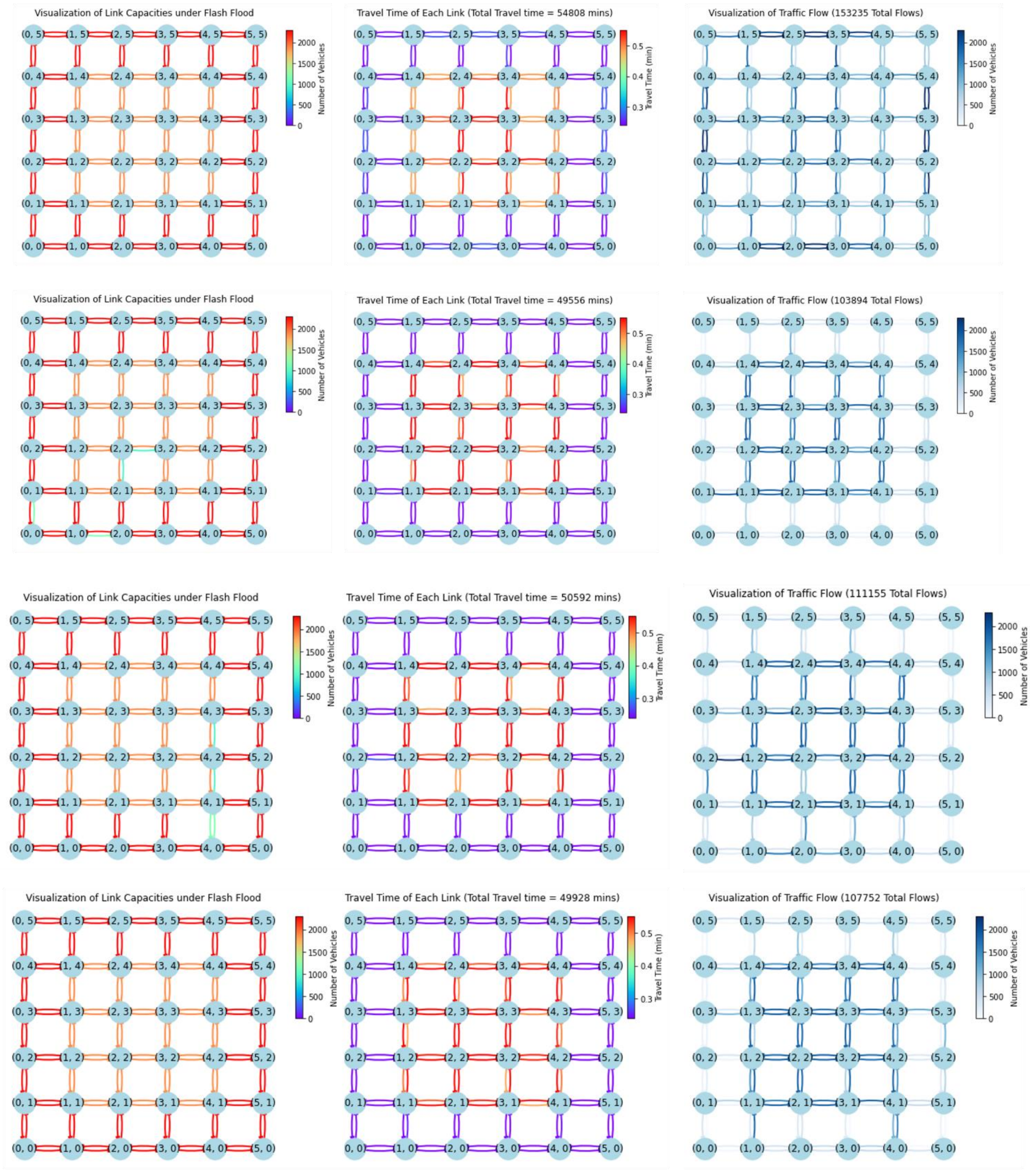


Figure 5.2A – part1: visualization of iteration 4-7

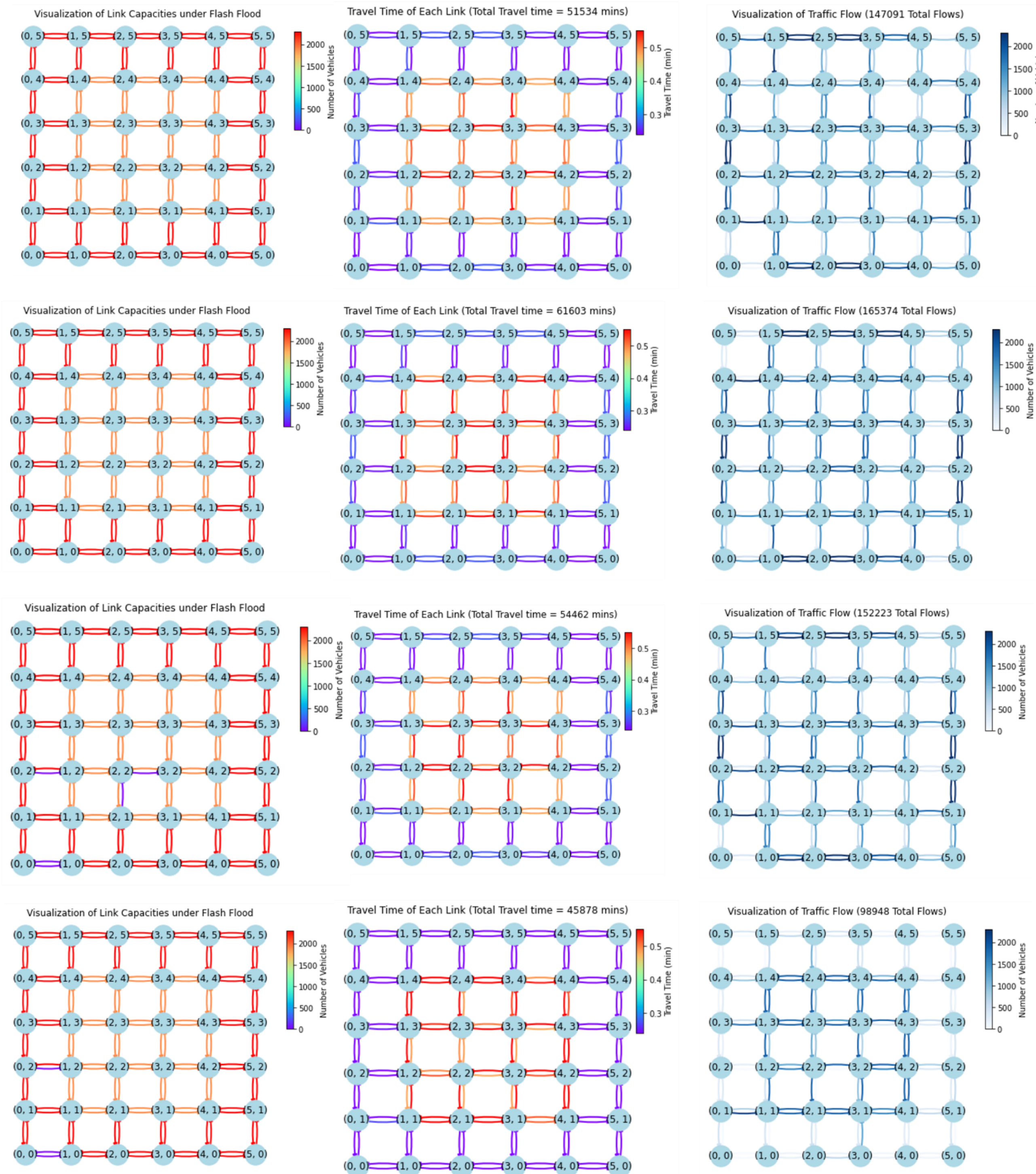


Figure 5.2A – part2: visualization of iteration 8-11

6. Conclusions

In conclusion, the DTA model developed in this project has effectively optimized traffic flow within the constraints of road capacity. It dynamically adapts to actual traffic conditions, maintaining overall travel time at an optimal level, notably lower than traditional routing methods that do not account for the impact of traffic flow. A key strength of the model is its responsiveness to varying flood scenarios. By rerouting traffic away from heavily inundated areas, the model effectively reflects the impact of flash floods through altered routes and travel times for the same OD matrix and network framework.

It is important to note that this project utilized a simplified, arbitrary network and an OD matrix, along with theoretical flash flood scenarios. Despite these simplifications, the model lays a solid foundation for future applications in real-world transportation systems. When integrated with a detailed road network and an actual OD matrix and coupled with a drainage model specific to the area, this model could assess the flash floods' impact on transportation networks. It holds the potential to provide optimal routing suggestions that are both practical and adaptable to varying weather conditions. The implementation of CBD and boundary nodes, along with the visualization functions, adds realism to the simulation as well. The ability to generate random OD matrices and introduce CBD-specific matrices enhances the versatility of the model.

Looking forward, an area for improvement in the current model is its approach to traffic assignment, which is currently handled for the entire trip from origin to destination simultaneously. A more nuanced approach could involve updating the routing as each batch of traffic arrives at a node, offering a more detailed capture of the network's flow dynamics. However, this would substantially increase computational demands. Implementing advanced algorithms would be essential for scaling this approach to larger networks, ensuring both efficiency and practicality in handling complex traffic patterns.

This project, therefore, not only demonstrates a successful application of dynamic traffic assignment under flash floods conditions but also lays a foundation for future enhancements and broader applications in urban traffic management, allowing for safer and more resilient transportation systems.

References

1. Choo, K. S., Kang, D. H., & Kim, B. S. (2020). Impact assessment of urban flood on traffic disruption using rainfall–depth–vehicle speed relationship. Water, 12(4), 926. https://doi.org/10.3390/w12040926.

2. Pregnolato, M., Ford, A., Glenis, V., Wilkinson, S., & Dawson, R. (2017). Impact of climate change on disruption to urban transport networks from pluvial flooding. Journal of Infrastructure Systems, 23(4), 04017015. https://doi.org/10.1061/(ASCE)IS.1943-555X.0000372.

3. Diakakis, M., Boufidis, N., Salanova Grau, J. M., Andreadakis, E., & Stamos, I. (2020). A systematic assessment of the effects of extreme flash floods on transportation infrastructure and circulation: The example of the 2017 Mandra flood. International Journal of Disaster Risk Reduction, 47, 101542. https://doi.org/10.1016/j.ijdrr.2020.101542.

4. Blöschl, G., Reszler, C., & Komma, J. (2008). A spatially distributed flash flood forecasting model. Environmental Modelling & Software, 23(4), 464-478. https://doi.org/10.1016/j.envsoft.2007.06.010.

5. Jamali, B., Bach, P. M., Cunningham, L., & Deletic, A. (2019). A Cellular Automata fast flood evaluation (CA-ffé) model. Water Resources Research, 55(6), 4936-4953. https://doi.org/10.1029/2018WR023679.

6. Kouroshnejad, K., Sushama, L., Sandanayake, H., Cooke, R., & Ziya, O. (2023). Pluvial flash flood-traffic interactions in current and future climates for the City of Ottawa. Safety in Extreme Environments, 5(3), 161-176. https://doi.org/10.1007/s42797-023-00077-5.

7. Li, M., Huang, Q., Wang, L., Yin, J., & Wang, J. (2018). Modeling the traffic disruption caused by pluvial flash flood on intra-urban road network. Transactions in GIS, 22(1), 311-322. https://doi.org/10.1111/tgis.12311.

8. Rebally, A., Valeo, C., He, J., & Saidi, S. (2021). Flood Impact Assessments on Transportation Networks: A Review of Methods and Associated Temporal and Spatial Scales. Frontiers in Sustainable Cities, 3, 732181. https://doi.org/10.3389/frsc.2021.732181.

9. Sheffi, Y. (1985). Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods. Prentice-Hall, Englewood Cliffs, NJ.

10. Peeta, S., & Ziliaskopoulos, A. K. (2001). Foundations of Dynamic Traffic Assignment: The Past, the Present and the Future. Networks and Spatial Economics, 1(3-4), 233-265. https://doi.org/10.1023/A:1012827724853.

Appendix: Division of labor

- **Topic selection:**

Atticus Rex; Garyoung Lee; Li-Yen Yang; Yifan Zhao; Flavia Kung'u

- **Literature Review / Project checkpoint 1 and 2 writing:**

Atticus Rex; Garyoung Lee; Li-Yen Yang; Yifan Zhao

- **Conceptual model structure, simulation idea, etc.:**

Atticus Rex; Garyoung Lee; Li-Yen Yang; Yifan Zhao

- **Main model programming:**

Garyoung Lee

- **Code wrap-up:**

Atticus Rex; Li-Yen Yang

- **Result analysis and report writing:**

Li-Yen Yang; Yifan Zhao

- **Report editing:**

Garyoung Lee; Flavia Kung'u

- **Video demonstration:**

Atticus Rex