Traffic Simulation

INFO 6205 Program Structure and Algorithms

Team #1

Repository

https://github.com/6205AF/FinalProject

Introduction

Boston is one of the most congested cities in the United States. I-93, one of the highways connecting Boston, Somerville and Cambriadge, plays an important role in daily commutes of the Greater Boston Area. Different vehicles from different cities merge on here. The goal of this project is trying to simulate the merge traffic where situations are similar to I-93.

In daily life, when a traffic accident occurs or the road surface needs to be repaired, the driveway will be temporarily unavailable. In addition, in order to meet the requirements of reducing the width of the road, the lanes will also be designed to merge. The project aims at simulating the traffic where merge happens. Through this simulation, we would like to get actual data and intuitive experience about traffic merging. Figure 1 depicts the image.

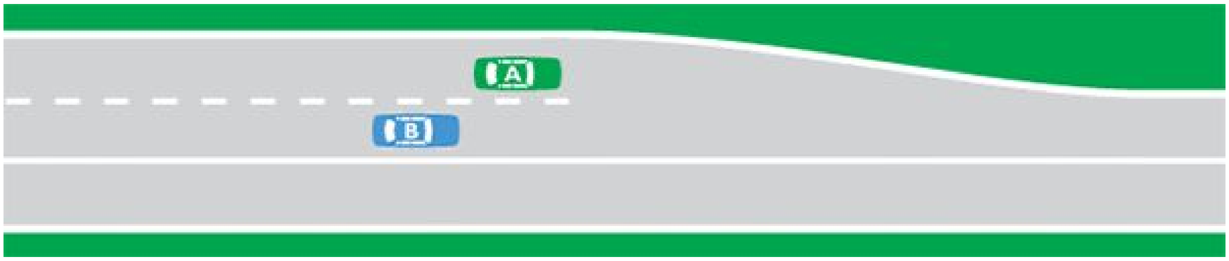


Figure 1. Lanes merge

Project Description

The project consists of three main parts. First is ***Vehicle***, objects of vehicles are given basic attributes such as position(px,py), vehicle length, speed ect. Second is ***Move***, which is trying to mimic every situation about cars moving in this merge traffic. For example, whether or not a car is accelerating; how cars can cut into the line and when to stop etc. Third part is GUI, implemented in ***Road***, ***Car***, and ***Truck*** with the support from ***Swing*** framework. They visually show what cars do. Also, the personally defined thread ***TrafficThread*** is used to manage the program status.

UML

In Figure 2, the project is conducted by the object-oriented programming language Java. With the critical trait of inheritance, all classes are designed as shown.

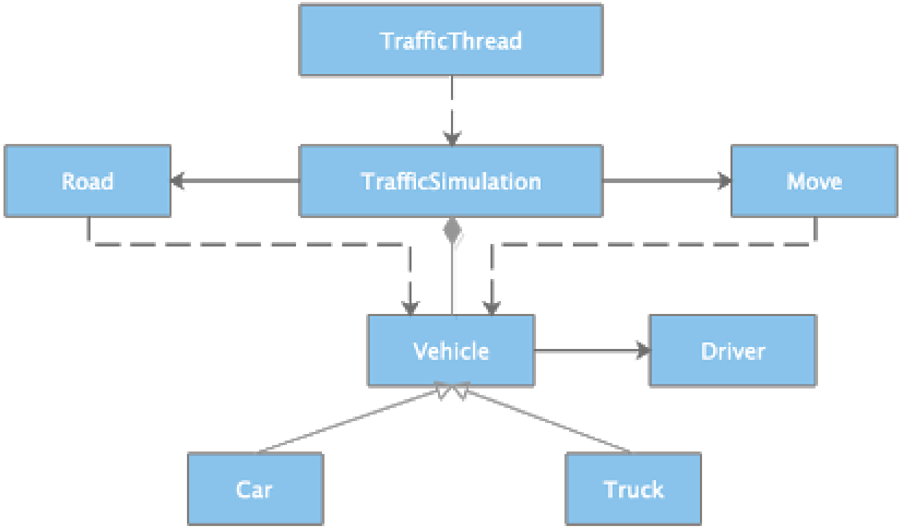
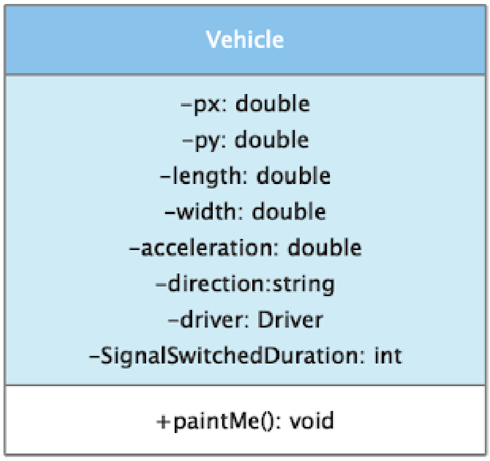


Figure 2. Class relationship

The structure of the core class ***Vehicle*** is as Figure 3.



Vehicle

-px: double
-py: double
-length: double
-width: double
-acceleration: double
-direction:string
-driver: Driver
-SignalSwitchedDuration: int

+paintMe(): void

Figure 3. Vehicle class

Design

In the project, three types of factors have been considered to simulate the real traffic situation. And it assumes the situation that five lanes merge to four lanes. The variants and constants are defined in ***TrafficSimulation*** class and can be utilized by other classes as static invariants.

- Variants
  - Stopping Distance
  - Traffic Density
  - Vehicle Length & Width
  - Driver Type & Possibility of Giving Way
- Constants
  - Road Condition: Dry
  - Distance to Previous and Next Interchange
  - Lane Width
  - Limit of Speed, Acceleration, & Deceleration
- Interruptions
  - Special Vehicle

Based on the assumption, the project simulates four situations: 4 lanes, 5 lanes, 4 lanes with special vehicles and trucks, and 5 lanes with special vehicles.

Output

Figure 4 demonstrates the situation of the program running. The four situations assumed are shown in the combo box.

Click the ‘Start’ button, and the simulation generates the fixed number of vehicles running on the screen. When one of them is willing to change the direction to left or right, the lights on the car head and tail will turn on to inform other cars in the road.

Click the ‘Stop’ button, and all cars become static because the thread is stopped.

The components in the north are Counter and Flow. Counter indicates the current number of vehicles in the graph. Also, Flow points out how many vehicles pass in 50 frames.

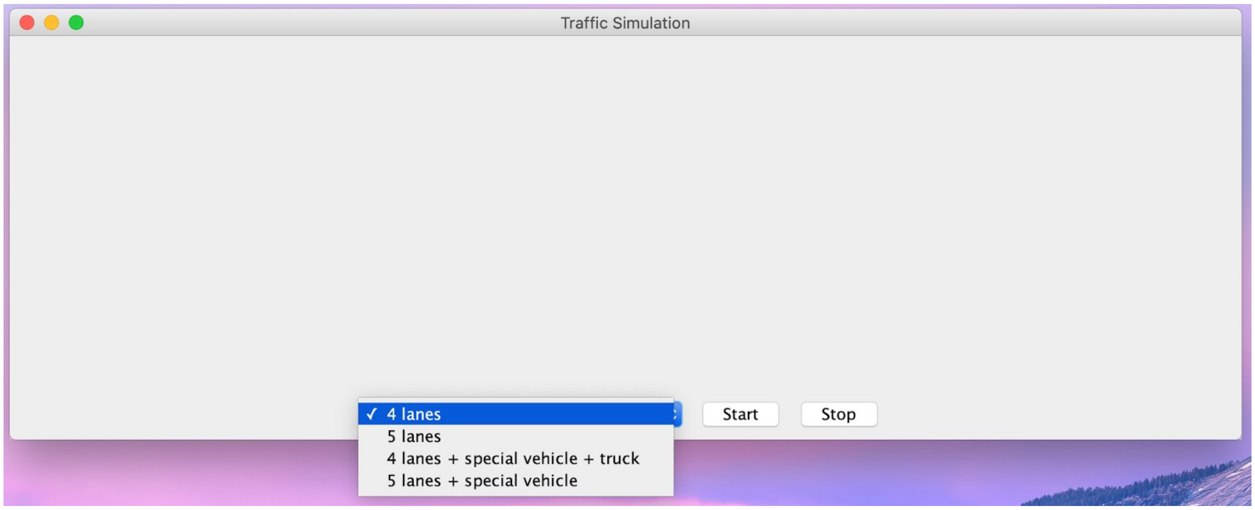
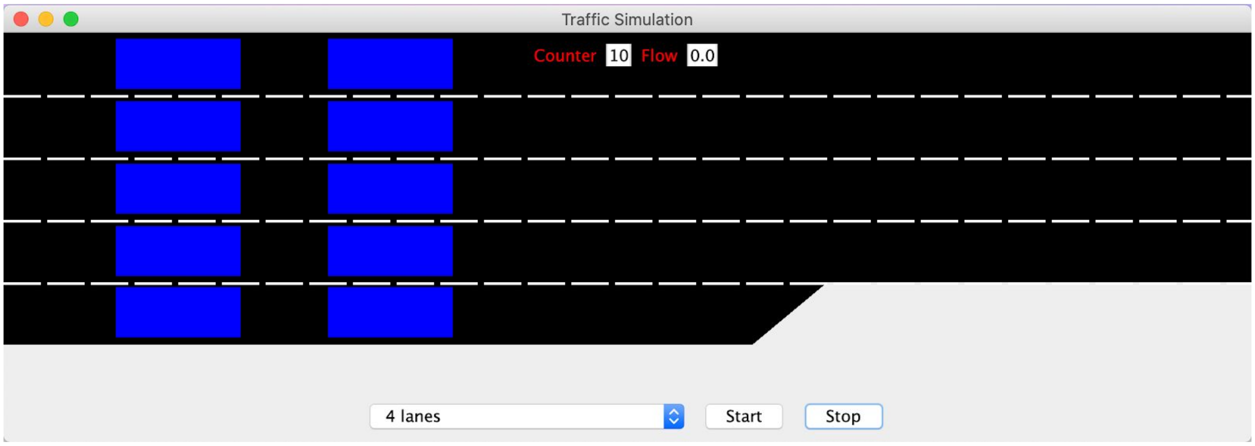


Figure 4. Overall Simulation for merging

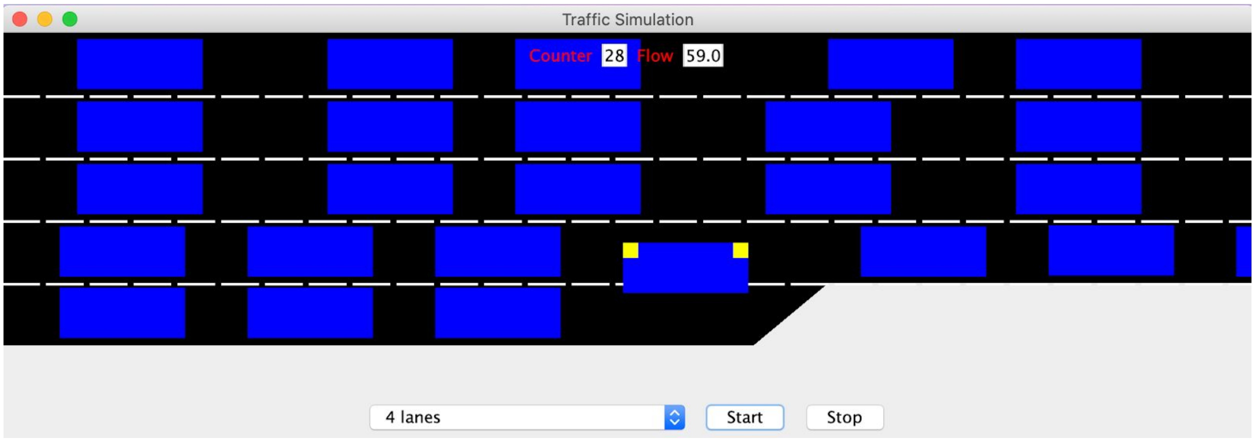
Figure 5 demonstrates the situation of speed limit in both situations. The speed limit for this highway is 100. If there is no car in front, the maximum speed is 100. If there is a car in front. The maximum speed is determined by the previous car and so on.



Counter 10 Flow 0.0

4 lanes

Figure 5. Maximum speed if no cars in front

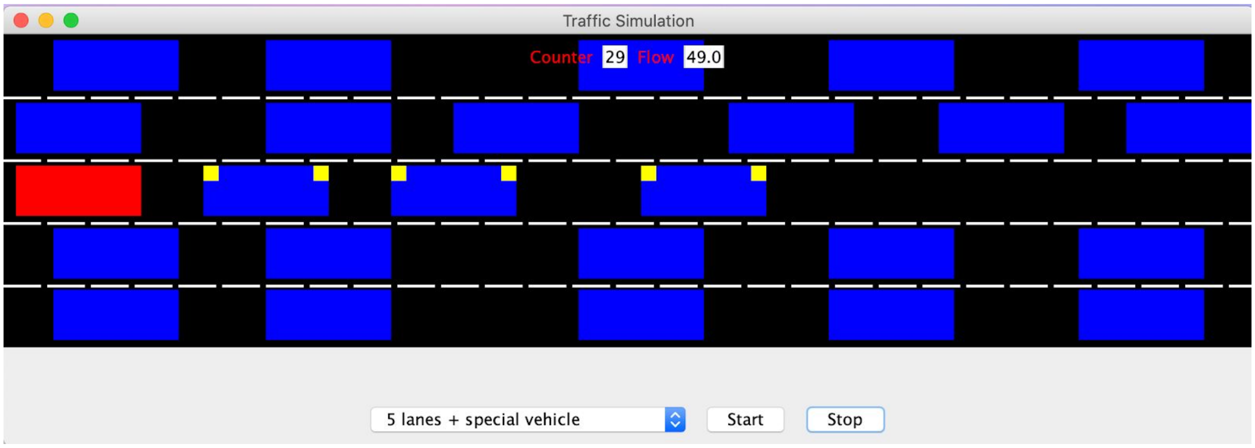


Counter 28 Flow 59.0

4 lanes

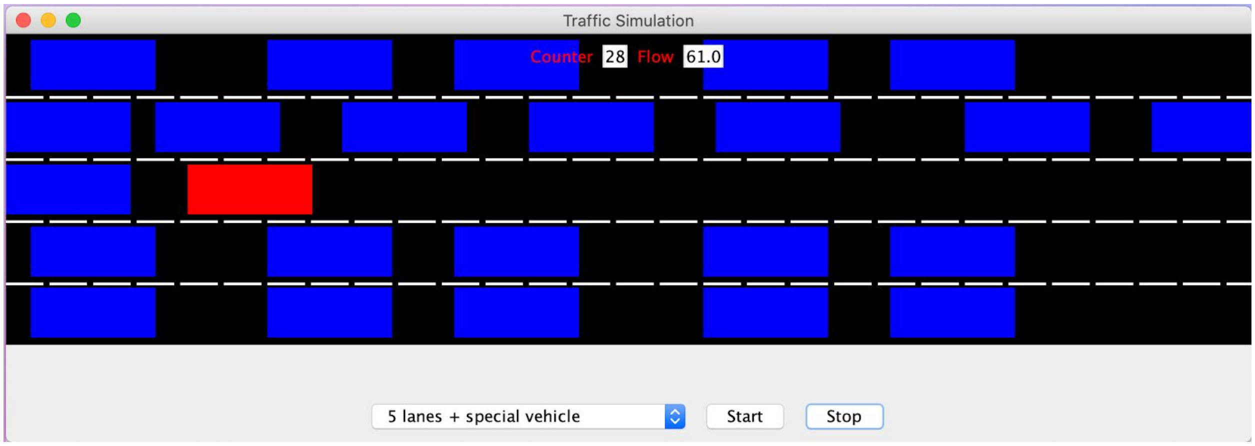
Figure 6. Maximum speed if there is a previous car

Figure 7 demonstrates what happens when a special vehicle is there. Other vehicles in front of special vehicles will either go left or right to clear the way.



Counter 29 Flow 49.0

5 lanes + special vehicle



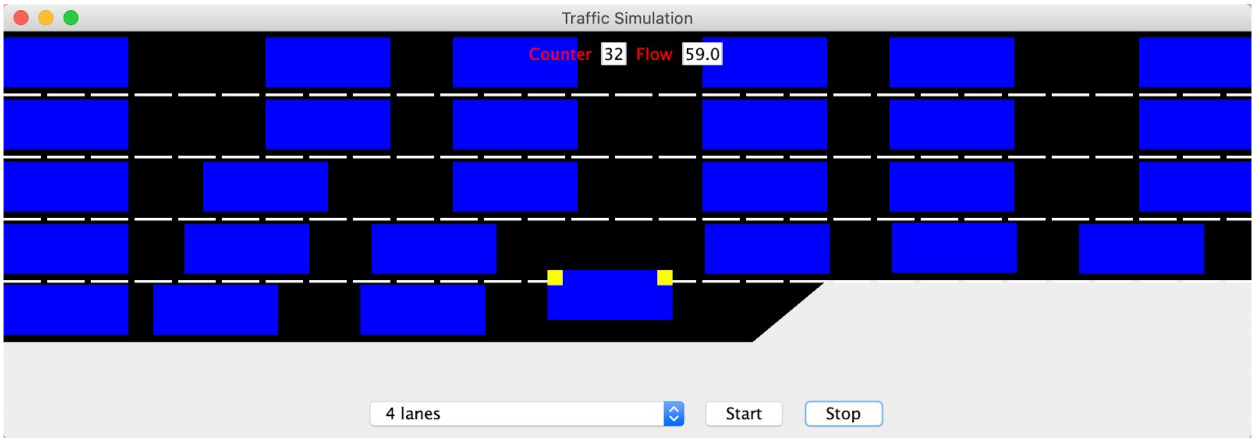
Counter 28 Flow 61.0

5 lanes + special vehicle

Figure 7. Special Vehicles

Four overall situations and flow comparison:

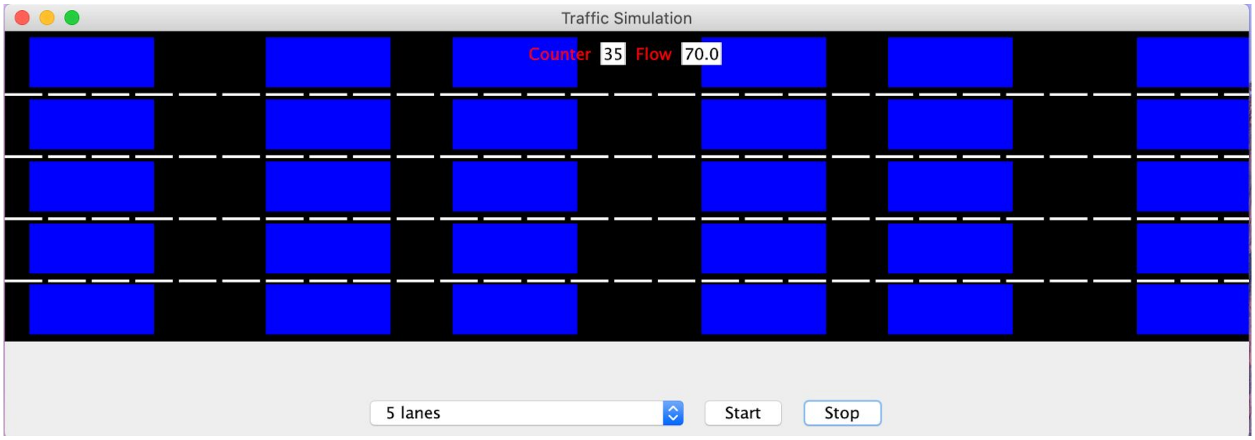
Situation 1: 4-lane simulation with merge but without special vehicle



Counter 32 Flow 59.0

4 lanes

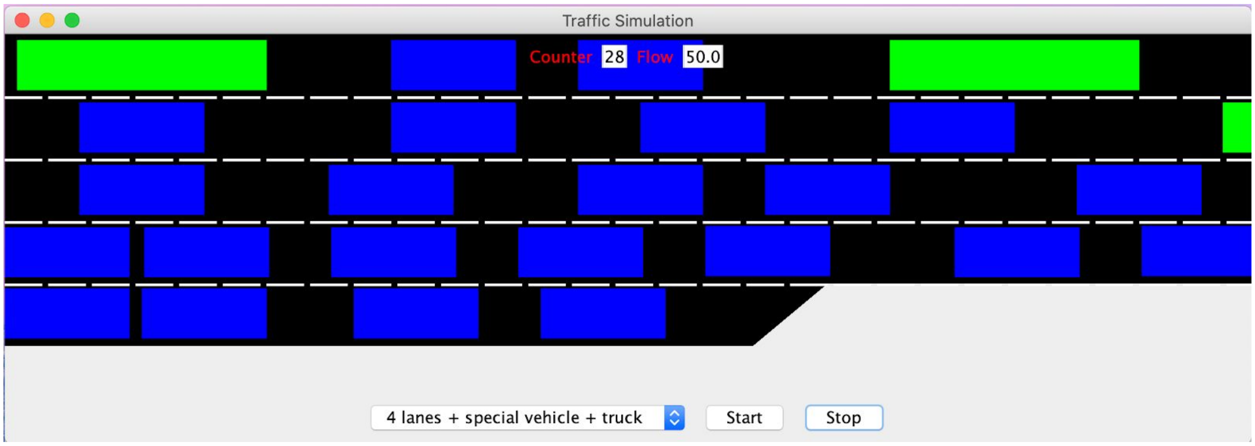
Situation 2: 5-lane simulation without merge and special vehicle



Counter 35 Flow 70.0

5 lanes

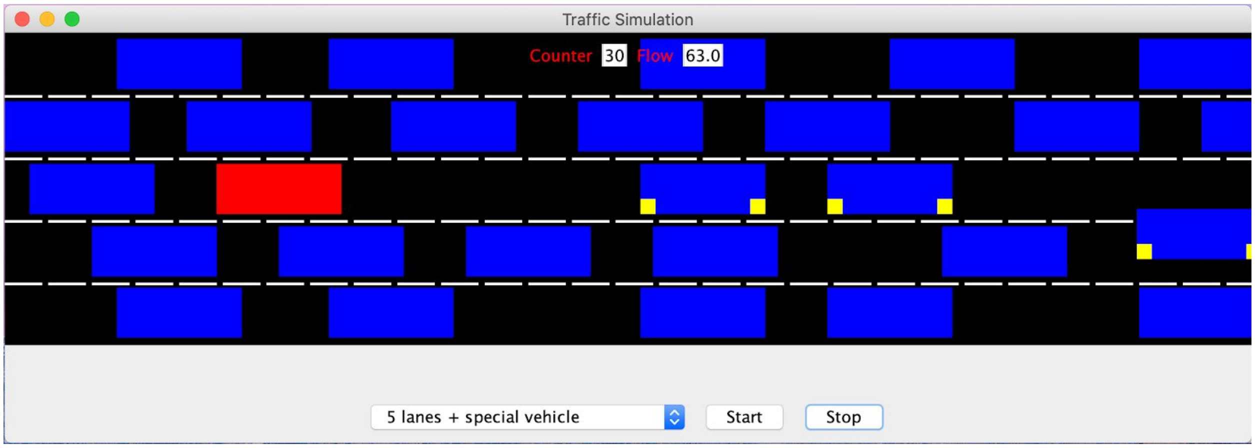
Situation 3: 4-lane simulation with merge, special vehicles, and trucks



Counter 28 Flow 50.0

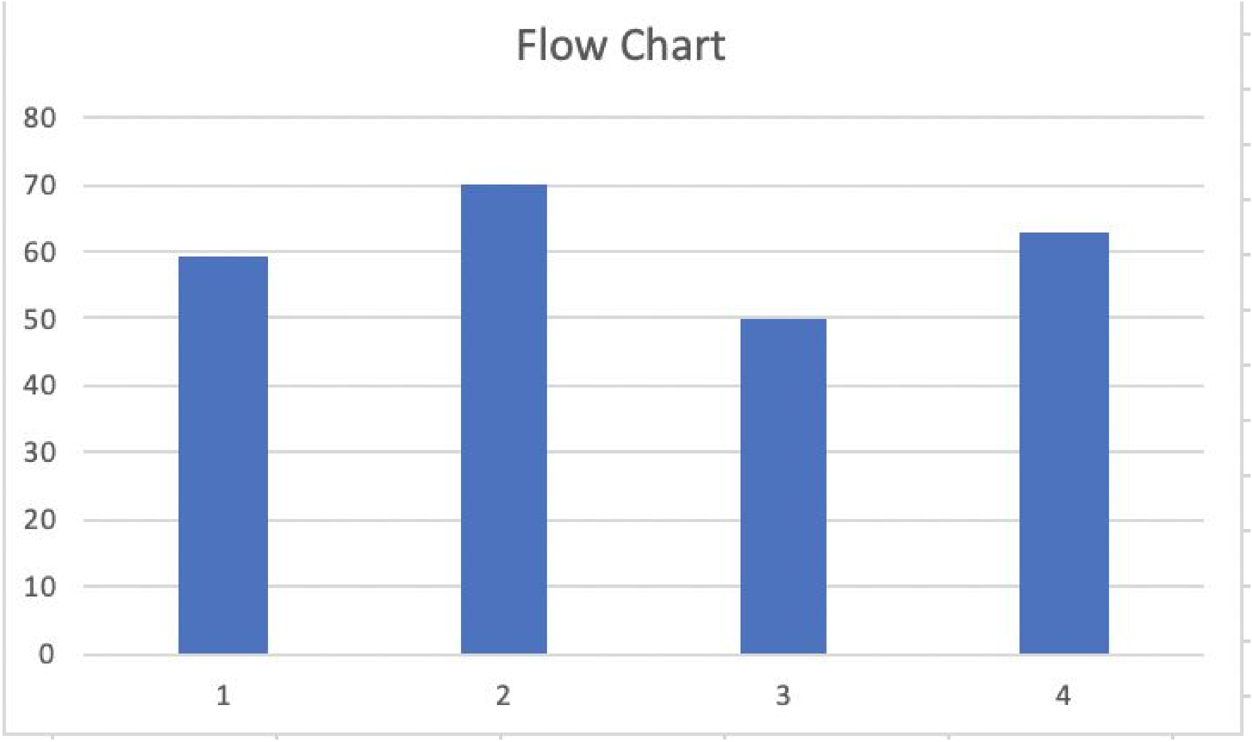
4 lanes + special vehicle + truck

Situation 4: 5-lane simulation without merge but with special vehicles and trucks



Counter 30 Flow 63.0

5 lanes + special vehicle



Conclusion

In normal, no need to merge situations, the flow is 70 to 75, which is highest among situations. When merge occurs, flow is reduced to 59. When we add special vehicles and trucks in a merge situation, flow is again reduced to 50. Finally, when we add only special vehicles in no merge situation, the flow is about 63.

Honestly, there is a difference among different situations but not significantly large. The reason may be because all these drivers are controlled by codes. Thus, they are extremely moving with order. In a realistic situation, it will be more chaotic and slower.