A Simulation of Manhattan Traffic System

Name:
Yifei Jiang
Kitty Li

Professor:
Charles S. Peskin

Modeling and Simulation in Science, Engineering, and Economics

October 28, 2022

Contents

1 Introduction

The state of traffic is closely related to people's daily life. In most cases, people care about the time spent on the road. In this project, we study how the average time of all the cars reaching their destinations relates to the rate of cars entering the system.

Considering real-life situations, the traffic system we set up includes one-way blocks connected by intersections, cars moving on the blocks, traffic lights at each intersection, and a navigation system. Our road map is set up based on a certain part of Manhattan shown in Figure 1.

To find a way to quantify the traffic system, we look at the average time cars take to arrive at their destination. When a car enters the system, the time recording starts. As the car moves along the blocks, it is guided by a navigation system that helps the car to choose the routes based on its departure position and the destination. When it reaches its destination and disappears from the system, the time recording stops. If a car does not reach its destination at the end of the simulation, we ignore that car when calculating the average time.

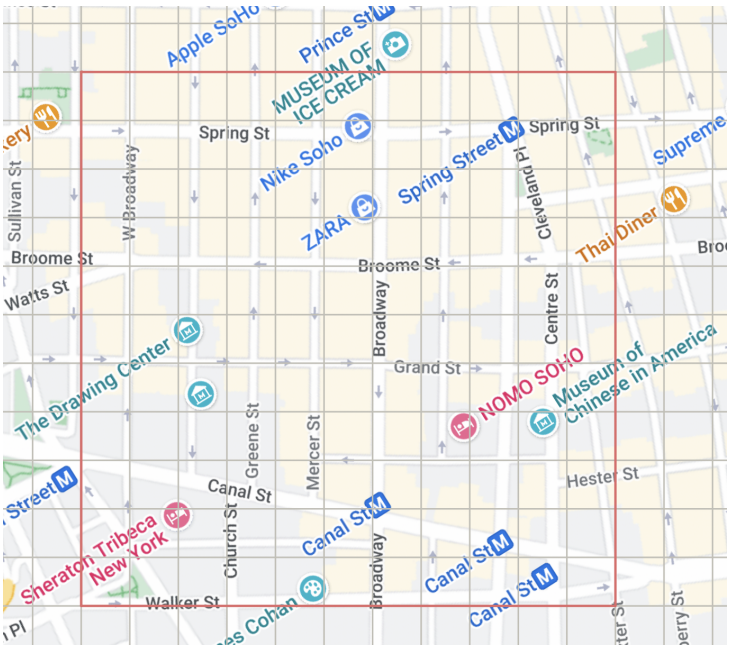

Figure 1: Traffic Network

2 Equations

2.1 Primary Equation

The most important factor determining the state of cars in the traffic system is car velocity. According to the car-following model in the microscopic traffic flow model,

we know that the velocity of a car depends on the state of the leading car, which is the closest car in front of it.

In our model, we assume the velocity v of a car only depends on the distance between it and its nearest barrier, which is either the car in front of it or the red traffic light. Also, we assume v to be proportional to that distance.

We define the velocity function in our model in the following way.

$$v = v_{max} \cdot \frac{d}{d_{max}} \tag{1}$$

Here v_{max} is a constant which represents the maximum velocity that a car can drive on the block, and d is the distance between the car and the nearest barrier which is no larger than d_{max} which is a constant.

2.2 Model Design and Secondary Equation

2.2.1 Traffic Network

The traffic network is made up of a collection of intersections and blocks. As shown in Figure 2, the circles represent intersections, the lines represent blocks and the arrows represent the directions of the blocks. By placing the network on a Cartesian coordinate system, we get the coordinates of each intersection.

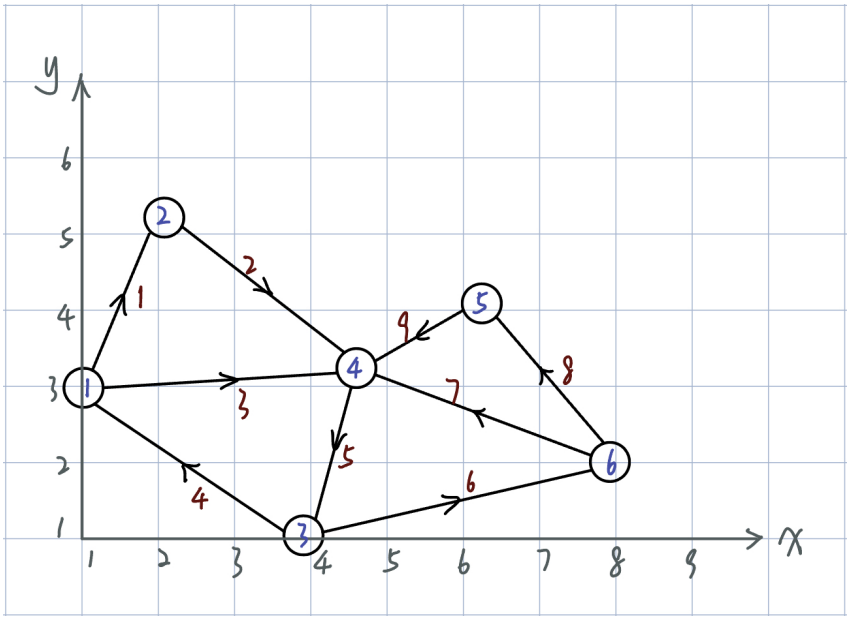

Figure 2: Sketch Map for Traffic Network

Every block starts from an intersection and ends at another. We define the starting intersection of block b to be $i1(b)$ and the ending intersection be $i2(b)$. Therefore, we

calculate the length of block b as followed

$$L = \sqrt{(x_{i2} - x_{i1})^2 + (y_{i2} - y_{i1})^2} \tag{2}$$

where x_{i1} and y_{i1} are coordinates of $i1(b)$, and x_{i2} and y_{i2} are coordinates of $i2(b)$.

Let x_u and y_u be coordinates of the unit vector in the direction of block b, then we have

$$\begin{aligned} x_u &= \frac{x_{i2} - x_{i1}}{L} \\ y_u &= \frac{y_{i2} - y_{i1}}{L} \end{aligned} \tag{3}$$

Let x_c and y_c be the coordinates of the car which is on block b, then we have

$$\begin{aligned} x_c &= x_{i1} + p_c \cdot x_u \\ y_c &= y_{i1} + p_c \cdot y_u \end{aligned} \tag{4}$$

where p_c is the distance between the car and the start of this block.

2.2.2 Navigation System

The Floyd-Warshall algorithm [1] is used for finding the shortest path between two points, with time complexity $O(N^3)$ and space complexity $O(N^2)$. There are only two possibilities for a path from node i to j. The first is that there is a direct path connecting i and j, and the second is that there are other nodes between i and j on the shortest path. Consider two matrices D and P. D is a matrix that stores the information about a system of nodes that we want to analyze, in this case, our road system.

To provide an explanation of the matrices D and P, we use the above Sketch Map (Figure 2) as an example.

We initialize matrix D by looking at each intersection index at a time. Obviously, the distance between an intersection and itself is 0, so $D_{ij} = 0$ when $i = j$. All the diagonals of matrix D should be 0. To construct a column i in D, we look at all the other intersections that one can directly go to from intersection i. For example, one can go from intersection 3 to intersection 1 and 6. Note that one cannot go from intersection 3 to intersection 4 because of the direction of that block. For all the intersections that cannot be directly accessed from intersection i, we consider the distance between them to be ∞. In matrix D, we can construct row 3 as $[\sqrt{13} \ \infty \ 0 \ \infty \ \infty \ \sqrt{17}]$, and column 3 stores the same information.

We do not have to initialize matrix P, since we will be updating P, the path matrix. The way we construct the path matrix is by first connecting all the intersections i j where one can go from intersection i directly to intersection j. Then, we use a nested for loop to determine whether a new path is shorter than the current path in which the

distance is given by the D matrix. After all the path in the P matrix being optimized, we have the shortest path stored in matrix P.

We know that any sub-path of the shortest path is still the shortest path. For example, the shortest path in the Sketch Map from intersection 1 to 6 is 1, 4, 3, 6. Then, if we want to know the shortest path from intersection 4 to 6, we have 4, 3, 6. The way we read the matrix P is by looking at P_{ij} where i is the starting intersection index and j is the destination intersection index. If i and j are directly connected, $P_{ij} = j$. If i and j are not directly connected, meaning one needs to pass some other intersection index k, $P_{ij} = k$.

In this way, we are able to obtain a shortest path given the starting intersection index and the destination intersection index.

3 Numerical Method

3.1 Creating Cars

The rule of creating cars in the traffic system is that, during each time step, we iterate all the blocks in the system, and for each block, if a randomly generated number between 0 and 1 is smaller than $(dt \cdot R \cdot L/(numbers\ of\ blocks))$, a car is inserted into this block.

3.2 Moving Cars

Since $\frac{dp}{dt} = v$, the position of the car at each time step is updated in the following way

$$p_{c_{new}} = p_{c_{old}} + dt \cdot v \tag{5}$$

When the position of the car is updated, we need to consider different situations. In Figure 3, there are four possibilities. Note that the position of the car p_c is the distance between the car and the start of the block where the car is currently on.

If the car has not crossed the intersection and the destination lies between $p_{c_{old}}$ and $p_{c_{new}}$ (the first case), this means the car has reached its destination and we remove it from the traffic system.

If the car has crossed the intersection, this means p_c need to be updated again by $p'_{c_{new}} = p_{c_{new}} - L$. If the destination lies between $p_{c_{old}}$ and $p'_{c_{new}}$ (the second case), the car has reached its destination and we remove it from the traffic system. If the destination has not arrived (the third case), we use Eq.4 to update the coordinates of the car.

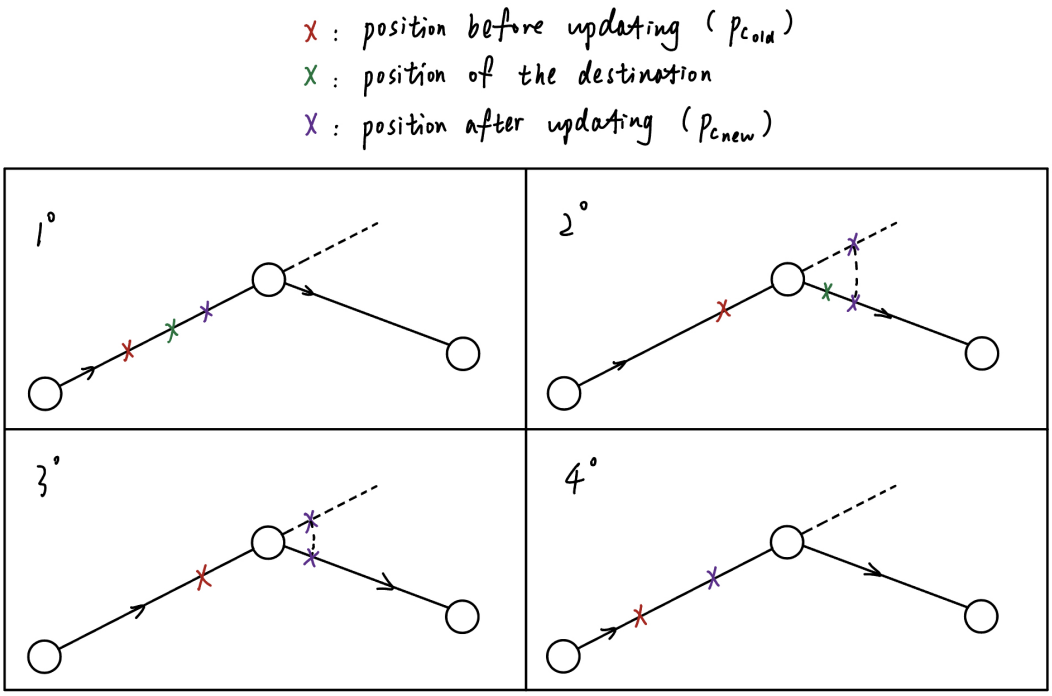

Figure 3: Four Possible Cases When Updating the Position of the Car

If the car has not crossed the intersection and the destination has not arrived (the fourth case), we use Eq.4 to update the coordinates of the car.

4 Validation

To validate our code, we need to check that when the time step dt is small enough, the result should not depend on it. We set the time of the whole simulation to be the same, which means $dt \cdot$*(numbers of simulations)* should be a constant.

As shown in Figure 4, we see that when $dt \leq 0.05$, the dots of results overlap a lot with each other. When $dt = 0.1$, the dots of results starts to go below the previous two results. When $dt = 0.5$, the dots of results do not make any sense because the dt is too large and the simulation cannot reflect the state of cars in the traffic system anymore.

5 Results and Discussion

5.1 Departure Position and Destination Random

According to the rule of creating cars stated in Section 3, we intuitively know that if the rate of cars entering the system R become larger, the average time of all the cars

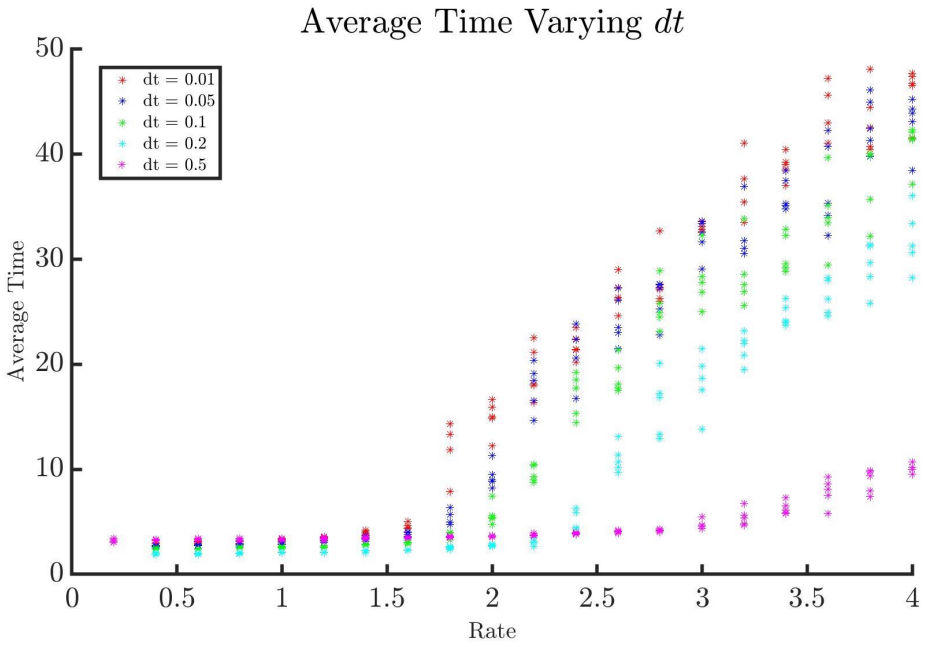

Figure 4: Result with Different dt

reaching their destinations will be longer.

By letting the $v_{max} = 6$, $d_{max} = 2 \cdot L_{max}$, $dt = 0.05$, and *simulation time*=200, we make a plot of Rate Vs. Average time and Rate Vs. Percentage of cars reaching the destination. Also, we ran the simulation for 15 times to obtain 15 data points for each R.

We find that the two plots are almost reflection symmetry. When $R = 4.4$, the points plotted begin to scatter, and when $R = 4.8$ the trends of the points start to go up for the average time and go down for the percentage.

Therefore, we divide the range of R into three parts.

The first part is $(0, 4]$ where almost all the cars created in the system reach their destinations and almost all the cars are driving at v_{max} since the density of the traffic is low. This leads to the fact that when $R \in (0, 4]$, the average time does not depend on R. (See Figure 6) (See the animation "randRequals1") [1]

The second part is $(4, 5]$ where the points begin to scatter and we call it the transition period. In this range, traffic jams may occur, but there exists a certain probability that they will not occur. At the same time, we see that the percentage of cars reaching destinations becomes lower as R increasing, which means some cars are inserted into the traffic system when the traffic jams have already happened and they fail to reach their destinations due to the traffic jam. (See Figure 7) (See the animation "randRequals5")

The third part is $(5, 10]$ where the points are all scattered and the trend of average time goes up steadily. The time when the traffic jam occurs becomes earlier as R be-

[1] For all the animations, the link is shown in the appendix

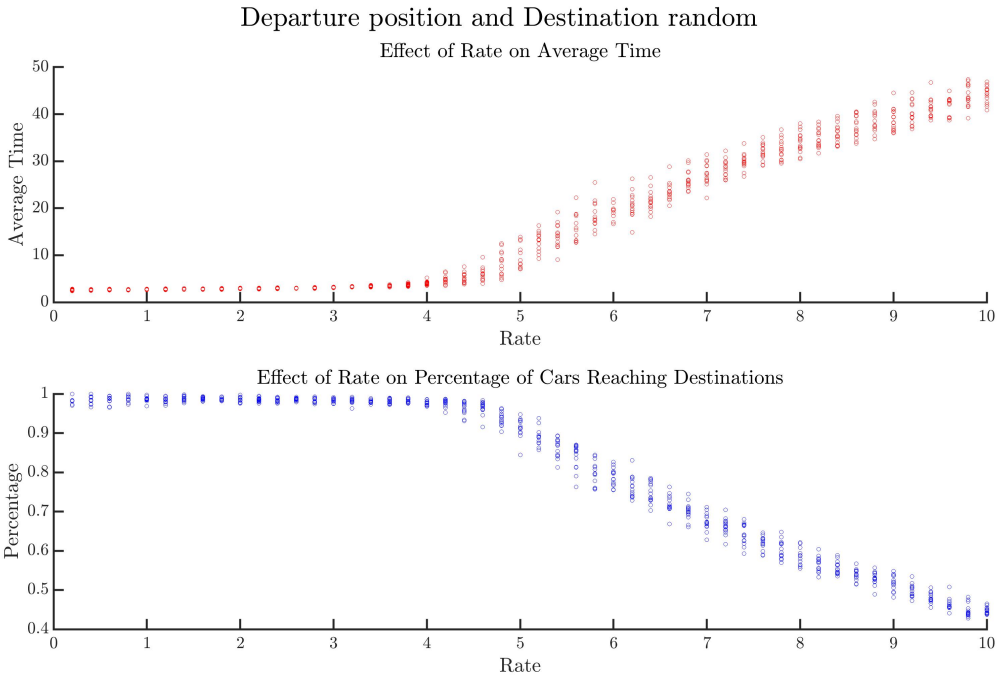

Figure 5: Departure Position and Destination Random

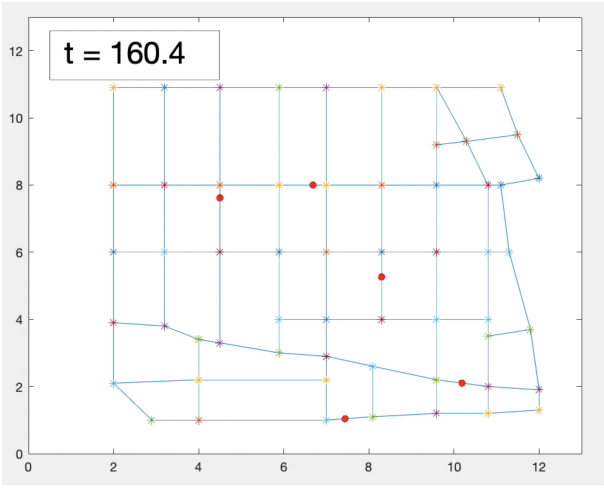

Figure 6: $R = 1$ Simulation

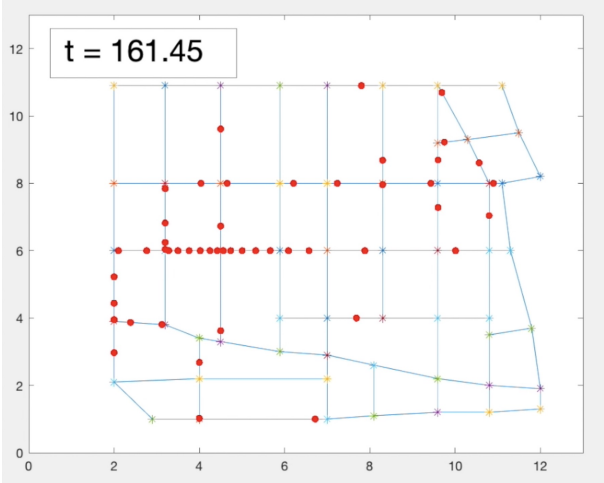

Figure 7: $R = 5$ Simulation

comes larger. (See Figure 8) (See the animation "randRequals9")

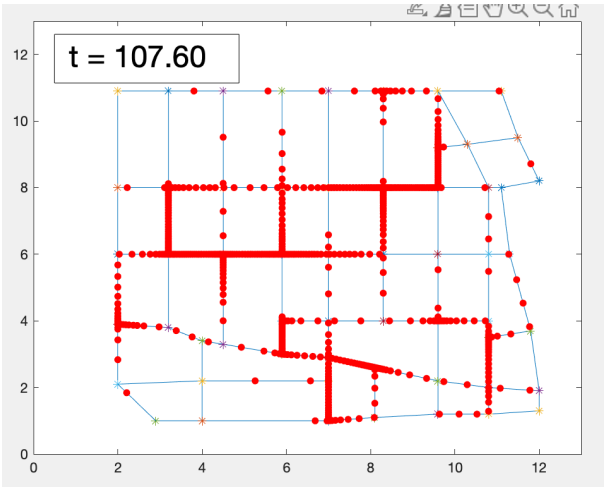

Figure 8: $R = 9$ Simulation

But we find that even though $R = 10$, the percentage of cars reaching destinations is still around 0.4, which is not too low. Therefore, we speculate that when R becomes even larger than 10, we will see the trend of the percentage approaching 0. Also, the trend of the average time will approach a certain value as well. We will verify this in the following analysis.

5.2 Fixed Departure Position or Fixed Destination

Instead of creating the car and choosing the destinations randomly, we let the car always be created in the same place or aim to arrive at the same place, which means they all have the same departure position or the same destination.

We let the departure position be fixed on the main block (shown in Figure 9), the departure position be fixed on the margin shown in Figure 10), the destination be

fixed on the main block (shown in Figure 11), and the destination be fixed on the margin (shown in Figure 12) respectively.

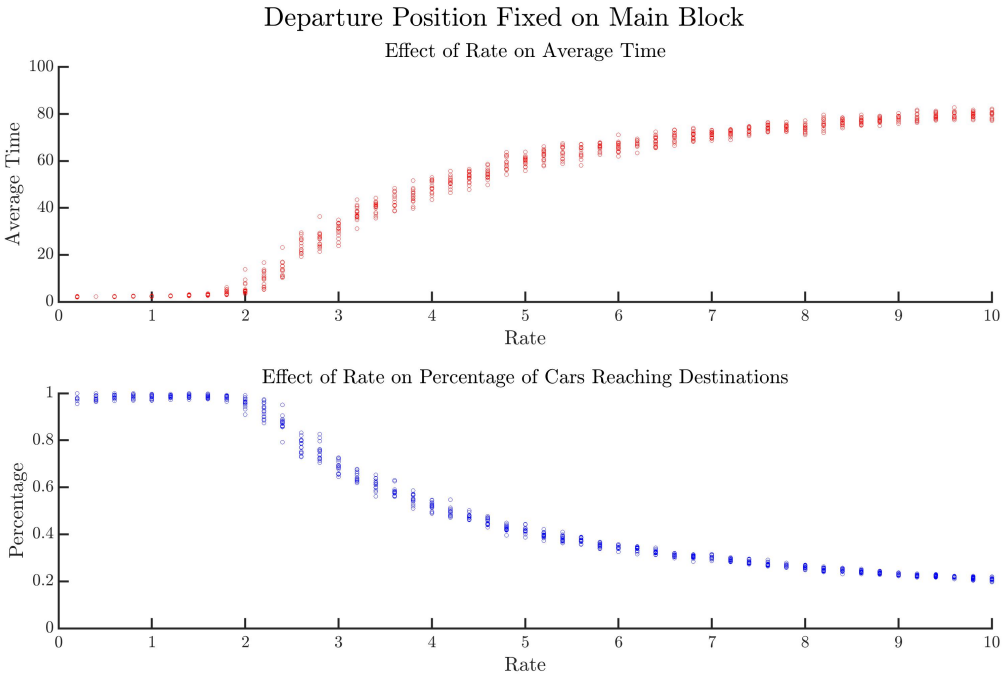

Figure 9: Departure Position Fixed on the Main Block

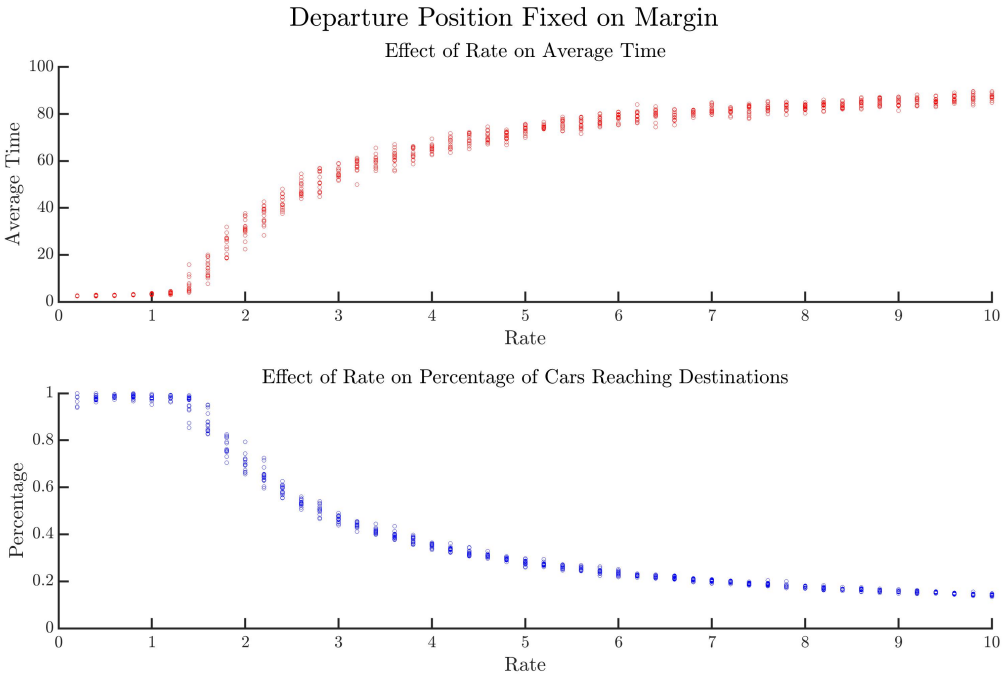

Figure 10: Departure Position Fixed on the Margin

We find that in these four cases, the transition period is achieved earlier than in the

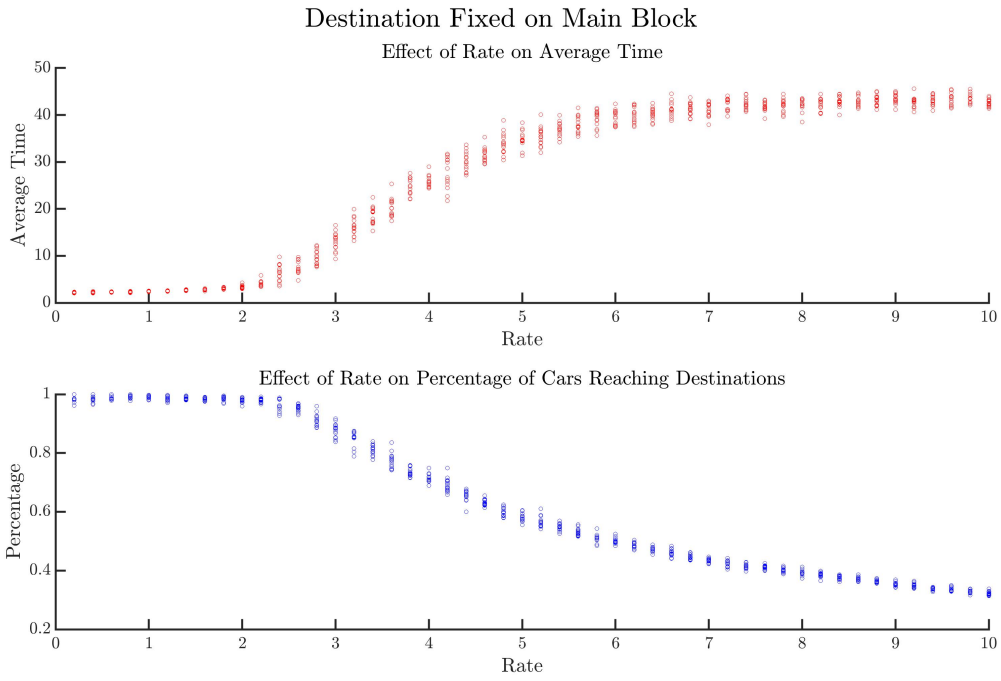

Figure 11: Destination Fixed on Main Block

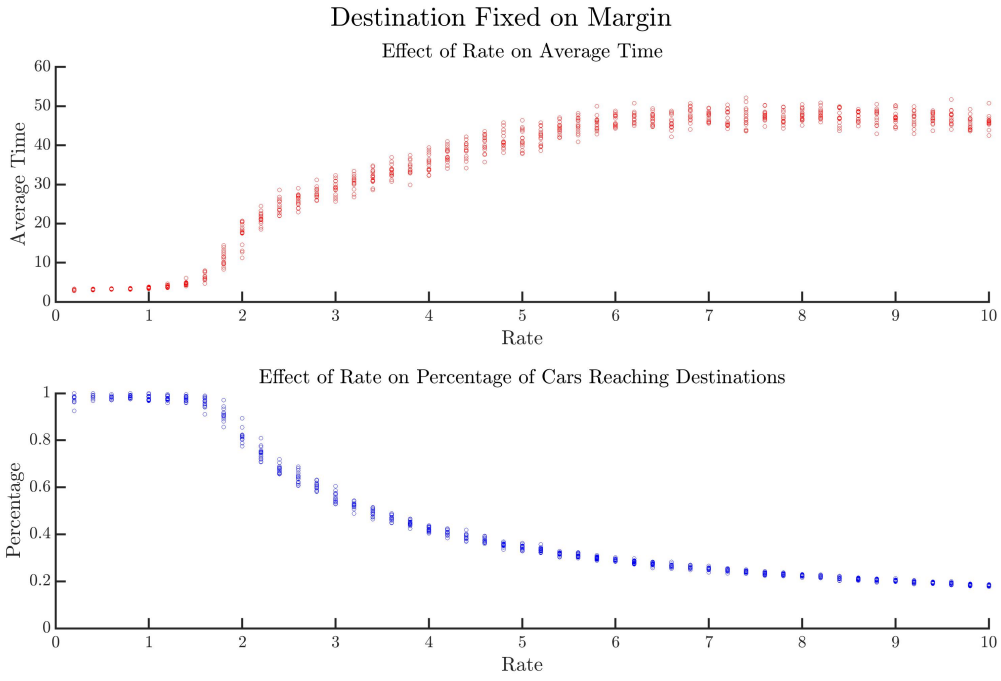

Figure 12: Destination Fixed on Margin

case where the departure position and destination are both randomly chosen. Also after the transition period, with R becoming larger, the slope of the trends for both average time and percentage becomes not as steep, which verifies our speculation that the trend of the percentage approaches 0 and the trend of the average time will approach a certain value.

More specifically, when the departure position or the destination is fixed on the margin, the transition period is achieved earlier than the cases where the departure position or the destination is fixed on the main block. That is because when a place is on the margin, the direction the car can choose to leave or approach it is much more limited.

When the departure position is fixed, the maximum average time being approached when R is large is larger than the cases where the destination is fixed. To understand why this happens, we focus on the difference between the two situations: when the departure position is fixed, cars are traveling to the same point; when the destination is fixed, cars are traveling from one point to other points. It is reasonable to assume that the traffic jam happens near the fixed location. Therefore, we can explain this by considering the effect of time. In the case where the destination position is fixed, cars arrive at the fixed destination at different times, thus less likely to result in a congestion than the case in which cars are generated at the same location at similar times.

5.3 Departure Position and Destination Random without Optimizing the Path

While in our model, cars navigate themselves by the shortest path, we want to verify that this approach indeed results in a shorter average time for the whole traffic system.

Therefore, we also plotted data we obtained from running a similar model[2] provided by Professor Charles Peskin (shown in Figure 13). In this model, while the departure position and destination are also randomly generated when adding a new car to the system, the blocks that each car chooses are not decided right after adding it. Which block to go is decided at each intersection by evaluating the vector from that intersection to the destination.

Being a reasonable navigation system, the average time is also increasing as the rate of adding new cars goes up. However, we can see that keeping everything else constant, the average time of this traffic system at R small is longer than that of our system. Therefore, it is fair to conclude that our model of cars traveling in the shortest path does result in a shorter average time for the whole system.

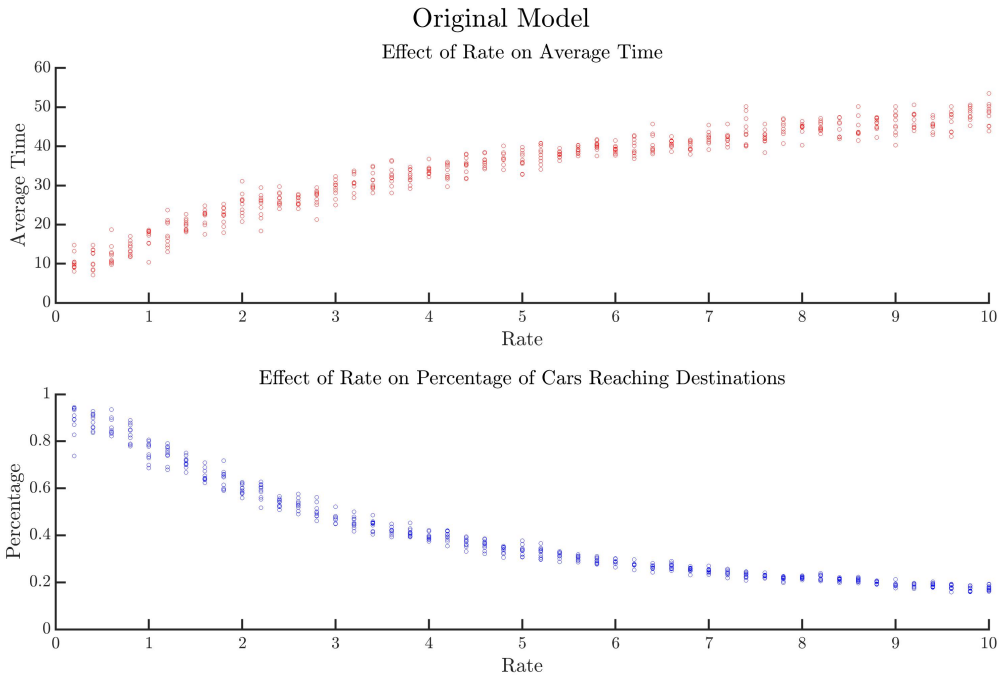

Figure 13: Model without Optimizing the Path

6 Summary and Conclusions

Our traffic model simulates the Manhattan traffic system on a limited map. While the situations for cars traveling on a real road are much more complex, our model provides a simple yet reasonable simulation of the traffic system.

In our main model, cars in the system are generated at different locations with random destinations, which is similar to the real life situation where people travel to different places in the city. With different rate of generating cars, this model successfully simulates the situations in which only a few cars are traveling on road at their maximum speed (for example, during midnight), and a lot of cars are traveling on road which results in a traffic jam (for example, 9a.m when people go to work).

A few special cases that we have looked at include people traveling to or from the same location in the margin or the center of the city. Those situations may happen in real life when people are traveling to an event and traveling back from the event. The simulation is also reasonably accurate compared to real life situation, which is also discussed in the section above.

While this simple simulation is reasonably modeled, there are some limitations. Future directions of this project may include considering a better velocity model for cars. Also, finding a path that optimizes the speed instead of the path.

References

[1] Shortest path floyd algorithm. https://learnku.com/articles/59637.

[2] Charles S. Peskin. Lecture notes of course modeling and simulation in science, engineering, and economics. https://www.math.nyu.edu/ peskin/modsim$_{l}ecture_{n}otes/index.html$.

Appendix for Links and Computer Code

Animation and Simulation Code for Traffic Model