CX 4230 Project 2 Final Report

Traffic Simulation

Jiayi Ye, Hanwen (Michael) Xu, Will Epperson

Github repo: https://github.gatech.edu/hxu317/ComputerSimProject2

CX 4230
Jiayi Ye, Hanwen (Michael) Xu, Will Epperson
April 23, 2019

Project 2 Final Report

Simulation Code

https://github.gatech.edu/hxu317/ComputerSimProject2

Problem Statement

Our simulation aims to enable the assessment of the average travel time for vehicles traversing a portion of Peachtree Street from 10th to 14th street in midtown Atlanta traveling in a southbound direction. Our simulation will have all the intersections present in real life, as well as all the traffic lights. We aim to examine light and heavy traffic, to simulate real life scenarios where certain times are more populated than the rest.

Conceptual Model

***Overview**:*
Our simulations consist of three models: event-oriented queuing, activity scanning queuing, and cellular automata models. These models share common assumptions and characteristics to ensure that our results are comparable. These common assumptions are elucidated below, while model-specific characteristics are discussed later.

Assumptions:
In our models, we assume that all cars are the same size and shape, simplifying the differences in vehicles. We assume that the cars all have the same max speed of 35 mph which is the speed limit on Peachtree. For a given level of traffic, cars are limited in their progress since traffic is more dense. At lower levels of traffic, cars will be moving faster, and vice versa for higher levels of traffic.

All of our traffic will be traveling southbound along the .4 mile stretch of Peachtree between 10th and 14th streets. We model this street as a two laned road where each lane is the same size and orientation. We have intersections with stop lights that simulate real life conditions at 10th, 11th, 12th, and 14th street. The timing of these lights is based on the real world timings provided with the NGSIM data. For our purposes, we model lights as a binary on/off switch where green corresponds to on and red/yellow correspond to off. The simulation started with default green traffic lights for all intersections.

Cars are able to enter our models at two locations: 14th and 11th street. Although cars can also potentially enter at 12th and 13th street, the NGSIM data only contained 2 and 8 cars

respectively entering Peachtree in the southbound direction at these intersections so we ignore these entrances. Additionally, we consider a car “out” of the corridor when it crosses the 10th street intersection and thus do not consider cars entering Peachtree from 10th. In calculating the average time to cross this corridor, we do not consider the cars entering at 11th street since they only travel a fraction of the corridor. Cars can only exit the simulation by crossing the 10th street boundary of Peachtree.

Inputs

***Generating Distributions*:**
To make our model more accurate, we used the NGSIM data to create empirical distributions of cars entering 14th street, cars entering 11th street, and the probability of lane changes for a car. For the distributions of cars entering, we first isolated which cars originate in the respective zone for each intersection (discussed below). We then find the minimum epoch time for each car that has this proper originating zone to find when they first enter the zone. With these times, we sort and then find the differences between the entrance times to generate our data. We then fit this data to a distribution so that we can randomly sample times of how often cars enter the respective intersections.

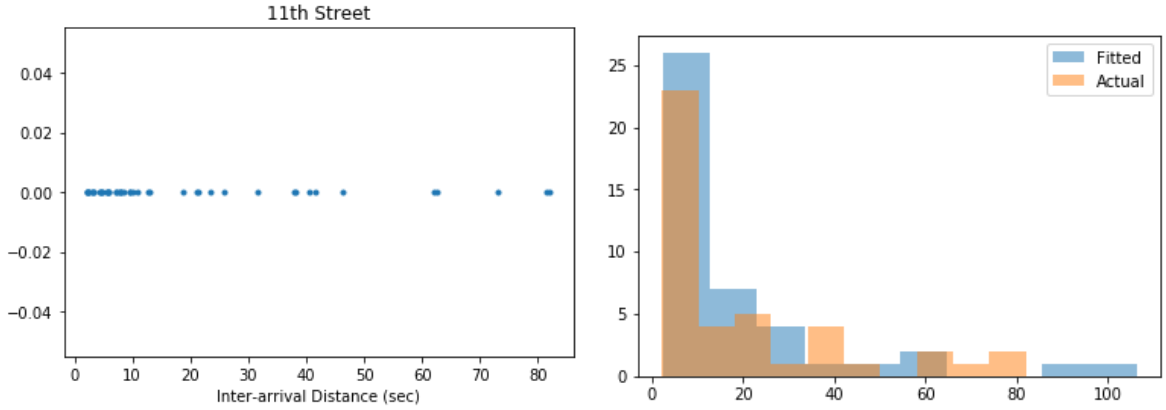
To select our distribution, we used the following procedure. Our candidate distributions were the normal distribution, lognormal distribution, gamma distribution, and exponential distribution. For each of these, we fit the data to the distribution using Python’s scipy library and then performed a Kolmogorov-Smirnov test to see goodness of fit. This test returns a distance metric as well as a p value. This p-value corresponds to our willingness to reject the null hypothesis that our data fits the given distribution. We select the distribution with the highest p value and also try to ensure that this p-value is above .05 (a common statistic threshold), thus ensuring that we fail to reject our null hypothesis and that our data fits the distribution well. The p values for each distribution are shown in the table below. For both 11th and 14th street, the best fitting distributions were the lognormal distribution. Although the p-value for lognormal the 14th street distribution was suboptimal, it was the highest out of all of the distributions and thus we used a lognormal distribution. This method of fitting to a distribution was based off a blog post made on insightsbot.com, and the blog post[3] is linked below in the sources.

Flow rate of traffic entering 11th and 14th intersections based on the NGSIM is considered as high because 4:00-4:15 pm is rush hour. The interarrival time generated from the fitted distribution is doubled to represent the low flow rate of traffic.

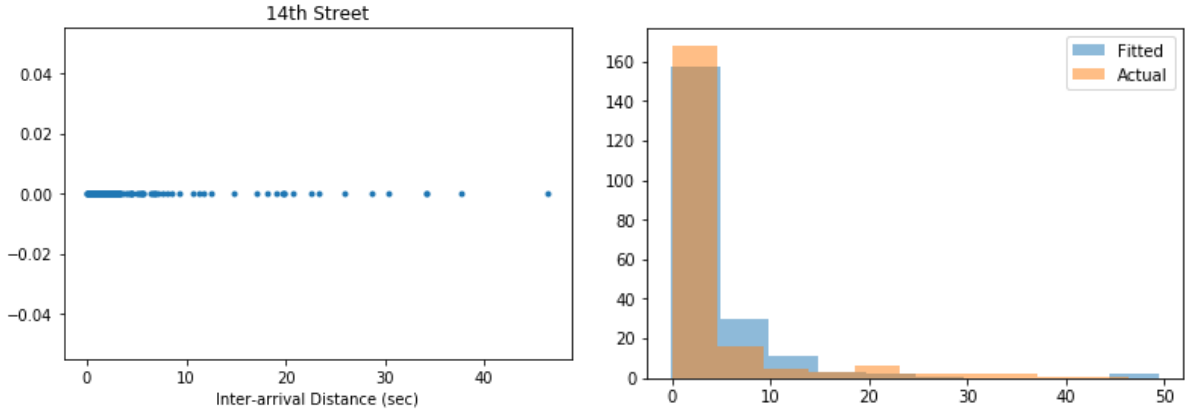
Distribution	**11th Street Data**	**14th Street Data**
Normal p-value	0.0054	4.27e-20
Lognormal p-value	**0.95**	**0.0042**

Exponential p-value	0.048	1.20e-10
Gamma p-value	0.20	1.43e-8

11th Street data distribution and sampled model fit:



14th Street data distribution and sampled model fit:



Distribution Parameters:

We used scipy stats library's Lognormal distribution, and sampled random values from it using the rvs() function. The parameters returned from our test were as follows.

Street	**Shape**	**Location**	**Scale**
14th Street	1.1235	-0.1308	2.2222
10th Street	1.5234	1.9716	7.6051

Distribution Verification:

Using the distributions above, we sampled to see how many cars would enter at each intersection for a 900 second run (or 15 min, the length of the NGSIM data) across 10 different

runs. For the 14th street distribution, there were 260 cars entering on average. Relative to the 207 cars contained in the dataset, this seems high however the distribution worked well in our models as discussed below. For the 11th street distribution, there were 31 cars entering on average. Relative to the 44 cars entering from this intersection in the actual dataset this seems low, however worked well in our actual models as well.

Lane Changes:

To model lane changes, we examined all southbound traffic and tracked whenever a car remained straight, made a change to a left-lane or made a change to a right-lane. These lane changes were detected by checking if a car's lane_id attribute changed between subsequent points. Using this method, we got the following probabilities of lane change (rounded to 3 decimal places).

- P(straight): 0.992
- P(left): 0.004
- P(right): 0.004

Traffic Lights:

Using the signal timing contained in the NGSIM data, we model traffic lights as a binary switch where on corresponds to green lights and off corresponds to yellow / red lights. Cars can only pass the intersection when the switch is on. The timing of these switches is below.

Intersection (Southbound)	**ON time (seconds)**	**OFF time (seconds)**
Peachtree and 10th	34	53
Peachtree and 11th	42	59
Peachtree and 12th	61	39
Peachtree and 14th	37	49

Simulation Models

Our simulation models were the following:

1. Event-Oriented queueing model: Hanwen
2. Activity Scanning queueing model: Jiayi
3. Cellular Automata: Will

***Event-Oriented Queueing*:**

To create an event-oriented queueing model, we first decided on a list of potential events that we would like to observe. In an event-oriented queueing mode, there is no need to model every time step, since we don't want to gather all the data at every possible timestep, but rather at important events. The events we chose are described below.

- Cars entering the southbound corridor from certain locations
    - In our model we can enter from 14th street and 11th street, so we have two events describing entering at either streets. The corresponding event handlers places the entering car at the corresponding intersection, waiting for the signal light to change.
    - In code, these events are named in the format: “Enter<StreetNumber>”
        - The corresponding event handler is in the format: “enter<StreetNumber>”
- Cars exiting the southbound corridor from the 10th street intersection
    - Since the only location cars can exit our model is at the 10th street intersection, we have one event describing exiting at this location. It’s event handler will keep track the exiting car’s total travel time if that car originates at 14th street.
    - In code, this event is named in the following format: “Exit<StreetNumber>”
        - The corresponding event handler is in the format: “exit<StreetNumber>”
- Traffic lights changing to red or green
    - In our model, traffic lights can only change to red or green, and there are traffic lights at the 14th, 12th, 11th, and 10th street intersections. There are two light changing events per intersection, one for the light turning red, and one for the light turning green. In total there are 8 total light change events.
    - When a green light event is processed, the corresponding event handler will schedule the next red light event at its intersection, and move all the cars waiting at its intersection to the next intersection. For example, the 14th street green light will move all cars waiting there to the 12th street intersection.
    - When a red light event is processed, the corresponding event handler will schedule the next green light event at its intersection.
    - In code, these events are named in the format: “RedLight<StreetNumber>” or “GreenLight<StreetNumber>”
        - The corresponding event handler is in the format: “red_light<StreetNumber>”, “green_light<StreetNumber>”
- Cars waiting at an intersection
    - In our event-oriented model, cars are not perceived to have constant motion but instead they move from a light to the next possible intersection, awaiting the next light change. Cars can only move on from a light when a green light is triggered, but not while a green light is on. This is a simplification specific to the event-oriented model. When processed, the event handler will add cars into the queue at that intersection, awaiting the next green light. The queue has no bounds for the total number of cars that can be at an intersection at a given time, so there was no attempt to model different lanes.
    - In code, these events are named in the format: “Wait<StreetNumber>”
        - The corresponding event handler is in the format: “wait<StreetNumber>”

***Activity Scanning Queueing*:**

Basic Concepts:

- Basic building block is the *activity*
- Model program's code segments consist of sequences of activities (operations) waiting to be executed
- An activity sequence's conditions must be satisfied before it is executed.
- Simulation executive moves from event to event executing those activity sequences whose conditions are satisfied

Car Movements:
Because activity sequence is important, three possible conditions need to be met and are used to trigger the car movements.
The action of the car moving is triggered when conditions are met:

- Car was generated
- Traffic lights: green

The action of the car stopping is triggered when conditions are met:

- Car was generated
- Traffic lights: red
- Distance of the car to the traffic lights is smaller than a threshold

Car Action	Condition 1	Condition 2	Condition 3
Car moves	Car was generated	Traffic light is green	
Car stops	Car was generated	Traffic light is red	Car is close to intersection

Simulation:

The activity scanning model runs the simulation for a total of 900s and changes the states of cars at each timestep of 1 second. Since the traffic lights on 13th street is not considered, the corridor is divided into 3 sections: 14th-12th, 12th-11th, and 11th-10th. The traffic lights are toggled according to the pattern of the real NGSIM data in each timestamp.

Cars are first generated on 14th street according to the fitted Lognormal distribution and put into two lanes with same probability 50%. Each car will be assigned an initial speed of 0 and initial position **x**. All cars will have a max speed with 35 mph and acceleration of 100*(speed limit - current speed). Along the southbound road, the cars will encounter the traffic lights on 12th street. The cars will move or stop depending on the conditions that check the traffic lights and distance to the next intersection. After checking the conditions, the car's probability to go straight to the next section is 99.2%. Following the same logic as in the first section, the cars will

travel along the section from 12th to 11th and from 11th to 10th street, and then exit after it reaches the 10th street intersection.

Cellular Automata:

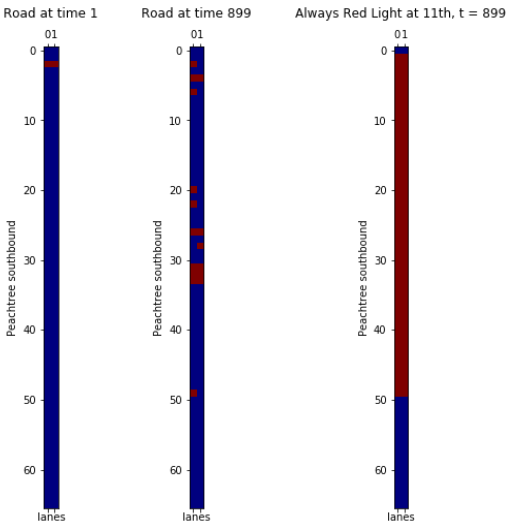
To create the cellular automata model, we used the following assumptions / model specifications:

Basic Model:

- Under the same assumptions as above, the corridor is .4 miles long.
- The average car is 16 ft long [see source 2]
- Typically, the space between cars is that of another car. Therefore each cell is 32 feet long.
- .4 miles = 2112 feet / 32 = 66 cells
- Since this is a two lane road, the world is thus modeled by a (66, 2) shaped matrix
- Each timestep *t* corresponds to one second

Movement:
All cars in our cellular automata model move at the same speed and are either moving or stationary. As discussed above, we cap speeds at 35 miles per hour. Since each timestep is one second, a car going 35 mph = 51.3 fps = 1.6 cells / t. Since we need integers for movement, we round this up to 2 cells per timestep (or a car going about 43.6mph). In any given timestep, a car will change lanes with the probabilities discussed above. Additionally, if there is already a car in the cell the car is trying to move to (either forward, or forward and lateral), the car will move as many cells as possible or remain stationary. This allows cars to queue while lights are red or traffic is heavy.

Traffic Lights:
Using the timings above, we added lights by picking rows that correspond to the geographic locations of these intersections and not allowing cars to pass this intersection when the light is off. These lights were placed at the intersections of 14, 12th, 11th, and 10th streets or after rows 0, 33, 49, and 65, respectively.



Visualizing CA:
To help with debugging and interpretation, we wrote some functions to plot the progress of the CA at different points. Above, you can see charts showing a sample run at time step 1 and 899. Cars are depicted in red and empty cells are blue. Due to traffic lights, clumps of traffic begin to emerge as seen in the model at time 899. When the traffic light at 11 is always red such as in the far right visualization,

we can see how cars queue up before this intersection. Since intersections do not actually take up in space in this model, they are not on the map.

Verification

***Event Oriented Queueing*:**
To verify the Event Oriented model, we tested each event handler separately. To aid in this process we implemented print statements that would be printed only during debug mode, which could be enabled by setting a flag in the code. Entering events were verified to ensure that the correct timestamp was given when first entering, and then verified that the car moved onto the correct intersection. Then at each subsequent traffic light, starting at 14th street, we checked the debug output to make sure the cars were moved to the correct intersection, and then waited at that intersection until the next light change. To verify the traffic light timing, we wrote a method to generate the full timing table for a given traffic light, then manually compared the timings given by the table to the timings given in the debug output. To test the overall system, we ran a test case with only two cars entering the system at time 15 and time 200, one from 14th street and one from 11th street. This way the two cars would meet up at the 11th street intersection and move on to the exit together. Stepping through this test run let us verify the traffic lights toggling correctly, and verify the correct movement of cars to the next intersection. We also tested the future event list, which was implemented with a priority queue using python's built in heapq implementation, and checked that every time we pop'd from the queue we would get the lowest timestamp object in the list by looking at the debug output.

***Activity Scanning Queueing*:**
In order to verify the activity scanning model, we tested whether the movement of cars will be triggered by one of the three conditions of the activity sequence. At each timestamp, print statements are implemented before and after each condition to see whether each condition is in effect and whether the car will move or stop when all the conditions are fulfilled. We first look at whether the cars that are generated will enter the model at the right time and whether those cars are going in the right direction. Then the traffic lights changing times are checked for accuracy based on the real world signal timings found in the NGSIM data. The distance of each car to the next intersection is checked using its current position and the end point of the current section. After all conditions are fulfilled, the cars will choose whether to move or stop, and this is again verified to be accurate based on the conditions it is presented with. To test the traffic lights and distance to the intersection, we put 1 car in each lane and track their speed and position along all intersections. For each section between 14th and 11th street, the previous conditions are printed to ensure the cars' actions follow expected behavior.

Cellular Automata:
To verify the cellular automata model, we performed several sanity tests to ensure the different parts of the code were working. Firstly, we tested that cars were moving properly by manually stepping through different time steps and looking at car movement. To test the validity of lights, we tried different configurations of lights such as leaving one always off like in the chart above and ensuring that no cars pass through. To make sure cars entering on 11th street were not included, we tracked car ids to only count the cars that entered on 14th and left at 10th in the

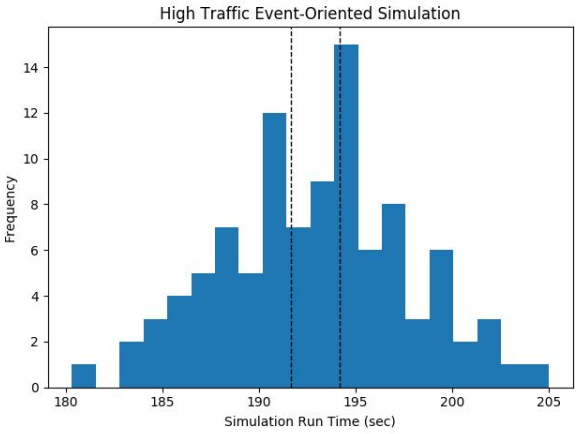
average. Additionally, we verified the results of the model by running it with all green lights and comparing this to a theoretical results. At constant speed, it would take a car traveling 43.6 miles per hour about 33 seconds to travel .4 miles. With our CA model with only green lights running for 900 seconds, it takes a car an average of 32.02 seconds to pass all the way through. This discrepancy of a second could be explained by the difference of counting or not counting the entry / exit cells in the distance traveled. Irrespective, the model is aligned with our expectation.

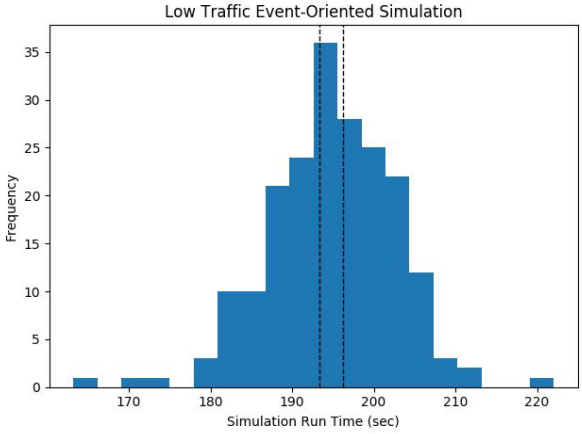
Results / Validation

From the NGSIM data, we were able to calculate a baseline average wait time of 133.96 seconds. This data was gathered by examining the cars that entered at zone 114, and exited at zone 201. These two zones correspond to the southbound entrance on the 14th street intersection and the southbound exit on the 10th street intersection.

***Event Oriented Queueing*:**

We performed a one-sample T-Test on the average wait times of each car passing through the southbound corridor that started at the 14th street intersection during high traffic and low traffic. Our null hypothesis is that our sample mean is a good estimate for our population of 100 iterations, and since our calculated p value is 0.999 for both high and low traffic, we do not reject our null hypothesis. This means our sample mean is a decent estimator for our population as generated by our simulation. The 0.99% confidence interval is also shown on the below histograms as dotted lines.

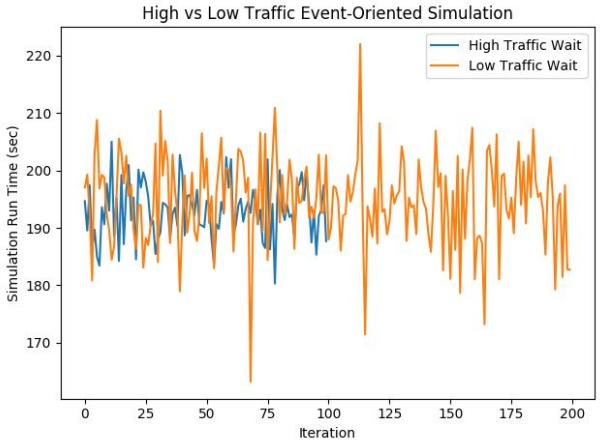




After running the event oriented simulation for 100 iterations, each with 900 timesteps, we found we had a sample mean of 192.923 seconds as the average wait times of each car passing through the southbound corridor that started at the 14th street intersection with high traffic. For low traffic, the average times don't change an appreciable amount, and the sample mean is 194.773 seconds. To simulate low traffic interarrival times, we doubled the interarrival times generated by the distribution we fitted using the NGSIM data. For both cases an average of around 190 cars passed through the corridor in the two simulations. The negligible difference in

wait times are due to a few factors, which also explain the inaccuracy of the model when compared to the baseline.

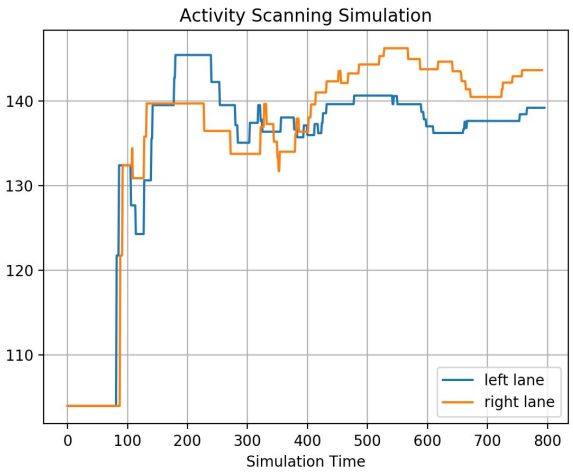
The first factor is the movement of cars in the event oriented simulation. When choosing events, we did not allow for the cars to move beyond the next light. In reality, the car would be able to move at the speed limit towards the next light, and if that light were still green the car would be allowed to proceed through the intersection. Compared to the current implementation of having to wait at each intersection, this change would undoubtedly decrease the average wait time significantly, as a car should be able to move through at least one intersection at the observed speed limit of 35 MPH.

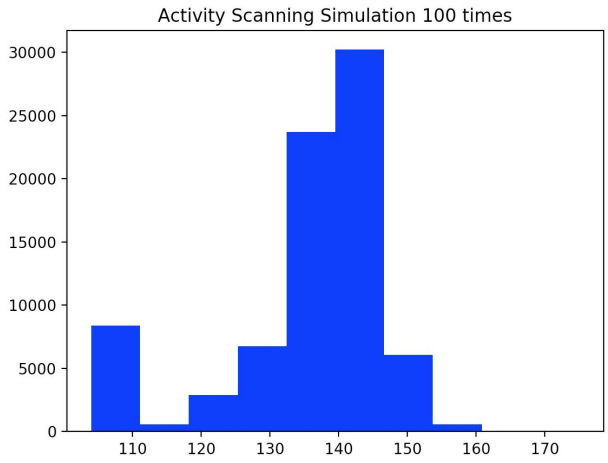


Another factor that should be taken into account is queue size at each intersection. In reality, there is a finite number of vehicles that can fit at each intersection, but in our simulation there is an infinite queue. This unbounded queue is what is causing the light traffic and heavy traffic cases to have very similar results, as this negates the difference between having a lot of cars and having very few cars. Changing how queues are handled would increase the average wait times due to the limited number of cars that can move through an intersection at a given time, but this increase would be negated by the decrease in wait time caused by the change to car movement stated above. Because of this infinite queue size, it should also be noted that there was no warm-up period, since the number of cars that could be in the road was unbounded.

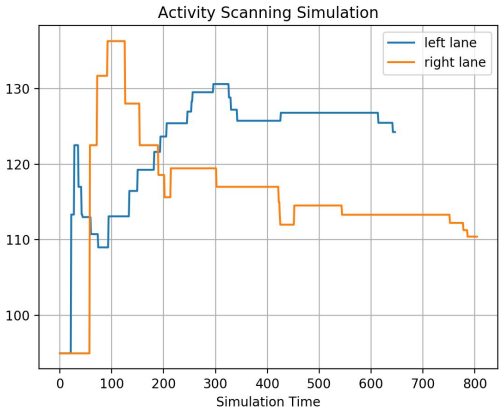
***Activity Scanning Queueing*:**

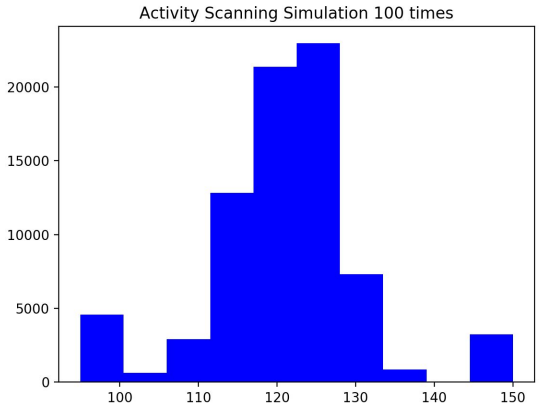
To see how the average running time changes overtime under a high flow rate of traffic, we first run one simulation for 900 seconds using the empirical distribution for interarrival times we generate from the NGSIM data. For each timestamp, we plot the average time of cars which have passed the 10th street intersection. We can see that the cars start to reach the destination from timestamp 90s onwards. After 200s of simulation, the warm up period ends and the average time a car spends travelling through the corridor from 14th and 10th stabilizes and the results are between 135s and 150s, as shown in the left figure below. After 100 runs of the simulation, the distribution of average running time is close to a normal distribution with confidence interval of $[137.115 \pm 12.803]$ for cars in both lanes, which is higher than the baseline 133 seconds.





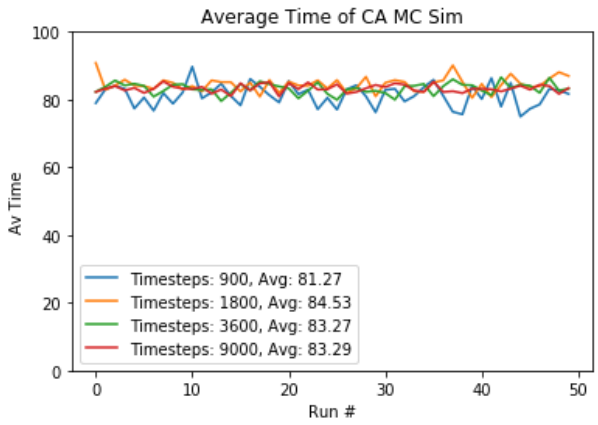
To simulate a lower flow rate of traffic, the interarrival time for the car generated on 14th and 11th street were doubled. In addition, the acceleration of the cars is increased by 10% to reflect the fact that with less cars on the road, each car would be able to accelerate more. The left figure below shows that the stabilized system has an average running time through the whole corridor of between 110s and 130s. After 100 simulations, the average running time is still close to normal distribution with confidence interval of [120.503 ± 10.409] for cars in both lanes. Thus, we can tell that the light traffic will result in lower travel time compared to the rush hours.

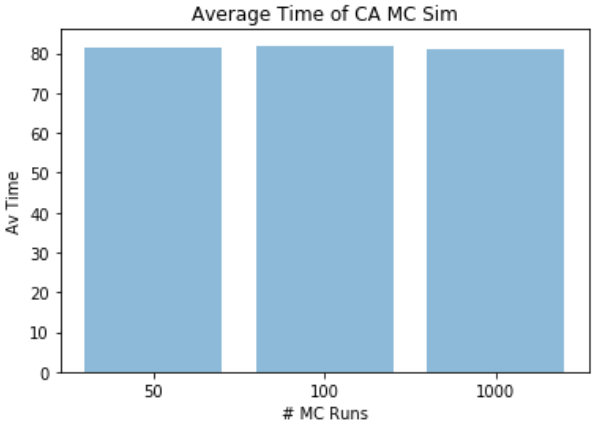




Cellular Automata:

Effect of Monte Carlo Hyperparameters:

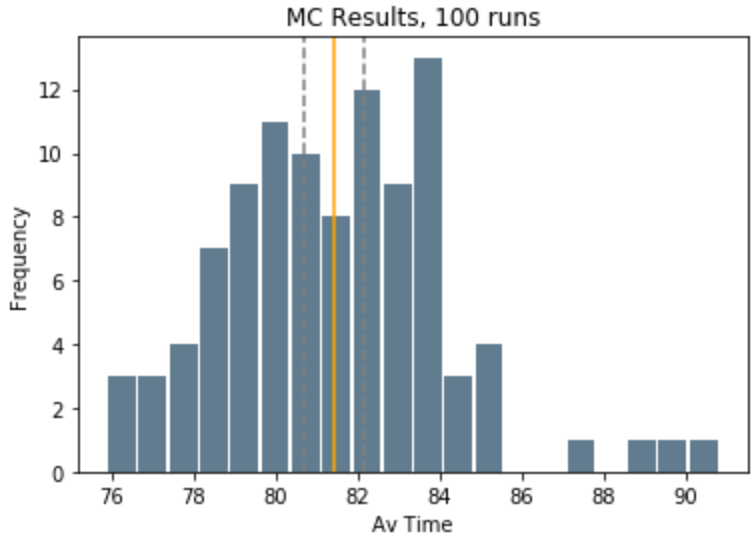




To assess the effectiveness of the Cellular Automata model, we ran Monte Carlo simulations with the model. As hyperparameters, we changed both the number of MC simulations being run and the number of timesteps the model ran for. These results are shown above. As the graphs suggest, the length of the simulation beyond 900 seconds or number of MC runs does not have a large impact on the simulation results. For instance, the mean times for different numbers of timesteps with 50 MC runs each only ranged from 81.27 - 84.53. Similarly, the mean time range for different numbers of MC runs with 900 time steps in each was 80.86 - 81.9. Keeping this in mind, we used 100 MC runs each with 900 timesteps for further analysis.

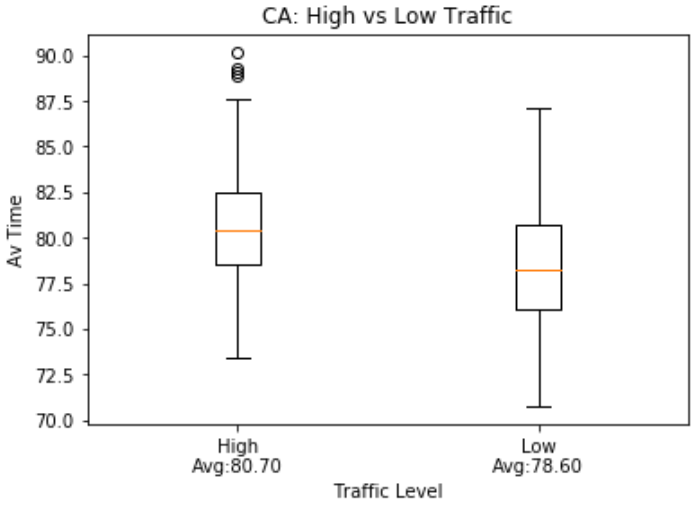
To get estimates of the average time it takes a car to traverse the entire corridor, enough cars must pass through to get an accurate results. The warm up period is the time in which either no cars have passed through the intersection or not enough to get an accurate estimate. With 100 trials each of 900 seconds, there were on average ~168 cars that passed through the corridor. According to the Central Limit Theorem and Law of Large Numbers, this distribution of average times should be approximately normal since 168 > 30 and thus we are confident 900 seconds is sufficient time for the simulation to run. Additionally, since the NGSIM data contains 15 minutes worth of data we consider 900 seconds a good simulation time.

Confidence Intervals for Mean Estimation:



The above histogram shows the distribution of average runtimes over 100 simulations, each lasting 900 seconds. The dashed grey lines show a 99% confidence interval for the population mean, and the orange line represents the sample mean. Our 99% CI yielded an interval of [80.715, 82.166]. Our distribution had a p-value of 0.999, meaning we would fail to reject the null hypothesis that our distribution is different than the global at a significance of 0.05 or 0.01. Since this distribution gives us a good estimate, we are confident 100 runs to be sufficient.

High & Low Traffic:



Additionally, we tested the effects of high and low traffic on the average time predicted with the CA model. As the above box and whisker plots demonstrate, the average time and general distribution is slightly faster for low traffic (a difference of ~ 2 seconds). Since we did not have real-world data from both high and low traffic settings we estimated cars to enter twice as fast during high traffic as low. Since our data was from 4:00pm - 4:15pm, we assumed this to be high traffic. We hypothesize that the small difference can be explained by two primary factors. First, even though cars enter with different distributions during high and low times, cars likely travel slower than 35mph in high traffic and thus would experience slower times in real world

settings. Secondly, the difference in distributions could be different than only a factor of two. We chose this factor based on our personal experiences in traffic, however real world data would be best to model both high and low traffic.

***Comparing Models*:**

Sample Means for Average Time to Cross Corridor (High traffic):

Model	**Sample Mean**	**99% CI**
Event Oriented Queueing	192.923	[191.65, 194.20]
Activity Scanning Queueing	137.12	[123.312, 149.918]
Cellular Automata	81.44	[80.72, 82.17]

As the data indicates, the event oriented queueing model has the highest predicted times, followed by the activity scanning queueing model and the cellular automata model. The cellular automata model is more than twice as fast as the event oriented model. Relative to our empirical baseline of 133 seconds, the activity scanning queueing model produces the most accurate results. These differences can be explained both by differences inherent to the models as well as assumptions within the cellular automata model. Since the cars must travel at least 1 cell, the car speeds in the cellular automata model were faster than in the queuing models. Additionally, cars do not accelerate from 0 in this model and thus move through the corridor more quickly.

Another point to note is the inherent difficulties of creating these different models. In the event oriented model specifically, it is extremely difficult to keep track of all the event handlers, especially when the number of events becomes too large. Scheduling events and event handlers also doesn't allow for easy look-ahead, which is what made it so difficult to calculate which light the cars could move to. This caused us to have to make specific simplifications for the event oriented model that further decreased the accuracy, and is what made it less accurate then another style of modeling, such as the activity scanning model, which came pretty close to the baseline measurement.

Potential Extensions

Across all of our models, several potential extensions could be made in order to better model real world data. On the whole, these extensions revolve around gathering more data or adding complexity to the models. As mentioned before, gathering more data from different times of day would allow our models to more accurately simulate traffic at any time of day. Gathering data at different times of the year would also allow us to account for potential seasonality differences. In terms of additional model complexity, in the real world cars enter and exit from more intersections. Adding these interactions to our models would let them be more widely applicable.

Further extensions include modeling different types of cars (large, small, motorcycles, etc.) and having cars with different speeds and accelerations.

Sources

1. NGSIM Dataset (provided on canvas)
2. https://www.quora.com/What-is-the-average-length-of-a-car-in-feet
3. http://www.insightsbot.com/blog/WEjdW/fitting-probability-distributions-with-python-part-1