CX 4230 Project 2 Final Report

Modeling & Simulation Life Cycle

Peachtree Street Traffic Simulator:

Generating the distribution of vehicle travel times as they traverse 10th to 14th street of Peachtree Street

Napa (Tan) Vananupong
William (Billy) Vaughn
Jaswanth (Jas) Sai Pyneni

Project Description

We are focusing on traffic on Peachtree Street from 10th to 14th Street. We want to simulate this corridor to understand average travel time of cars in relation to levels of traffic. Furthermore, we want to compare different simulation approaches to this problem to understand the methodology and the necessary code development. The project can be found at: https://github.gatech.edu/wvaughn9/PeachTreeSims. There is instructions in the readme to run each model.

Conceptual Model

We will be simulating traffic traveling from 10th to 14th. We are only simulating traffic in one direction. We will be basing the simulations off of the entrance time and total time in the corridor. We will be analyzing the distribution of vehicle travel time in relation to levels of traffic, as measured by vehicle arrivals per minute. For the purpose of this model, we are simplifying to using the same speed for each car and we will explain later how we decided that speed. In the following section, we will also explain how we assessed the random timestamps for new travel events. The three model we will be using are: Event-oriented queueing model, Cellular automata, and the Activity Scanning model. We will build an Event-oriented queueing model and refactor that into an Activity Scanning model to understand how to approach modeling with these two similar methods and then do a Cellular Automata model to test with a different approach as well.

Assumptions

- Only modeling traffic in one direction
- When traffic lights are green, cars can pass through the intersection
- When traffic lights are red, cars cannot pass through the intersection
- Traffic lights are either red or green, based on the NGSIM data light timings (yellow counts as green)
- Once cars have crossed 14th street intersection, they exit the world view
- Cars can enter at 10th and exit at 14th, no other entrances or exits are permitted
- All traffic lights start off as green
- There will be 2 lanes

Event-oriented Queueing Model (intersection by intersection)

a. Arrival to intersection
b. Traverse intersection (cross through)
c. Departure from intersection

I will model this corridor intersection by intersection, with the above three events as handlers that handle the events of traversing an intersection. All the events will get added to the same FEL.

a.) The arrival to intersection event handler checks if the car arrived when the signal was green or red and if it is during red, schedules the next pertinent event for that car further away (the remaining time of the red light away) as opposed to right away if it was green.
To check if the car arrived during green (or yellow) or during red, the following code is used:

```
g,y,r = signal_time_NB[curr_inter] #signal times for 10th st light
signal_cycle = g+y+r
wait =  0
if ((now % signal_cycle) > (g+y)): #arrived when red light
    wait = r - ((now % signal_cycle) - (g+y) )
    waiting_counts[curr_inter] = waiting_counts[curr_inter] + 1
```

In this, there is a signal cycle that is calculated as the sum of the duration of each of the green, yellow and red lights for the Northbound signal at each intersection. By modding the global time with that signal cycle, I calculate at which point in the signal cycle the car arrived. If it's past the duration for green and yellow, then it arrived during red signal and will factor in having to wait the remainder of the signal before scheduling the crossing of that intersection. If there is more than one car waiting for the green signal at that intersection, then I added half the intersection cross time to calculate the timestamp of crossing event to model that a waiting car cannot cross the intersection until the previous car is halfway through the intersection.
In the arrival handler, for every car that arrives to the 10th street intersection, I schedule a new car to enter the 10th street intersection (and so the whole system), as long as the total car count is within a specified cutoff, by assigning the new event a timestamp that is randomly generated using exponential distribution around a specified mean inter-arrival time, which will be explained later.
b.)The crossing event handler handles finishing the traversal of the intersection and schedules the departure event to leave the intersection.
c.)Since we are modeling entering the corridor from 10th street and exiting at 14th street, my departure handler will schedule the entry into the next intersection, after traversing the subsequent section of road, for all intersections before 14, and departure from the 14th will trigger some global statistical calculations. In the my final model, I simulated the sections between the intersections as one strip of movement after crossing the intersection. I simplified it and modeled it such that the sections of the corridor do not have lanes and cars cannot take over each in that space. This is because all my cars move at the same speed and I wanted to simplify everything and study the throughput of cars through the corridor as affected by just the level of inter arrival times. This model can be run by going to the Event_Oriented_Jas directory and running "python simulationengine.py"

Cellular Automata

New cars can also only enter when the 10th street light is green since this is the start of my model. The cars follow the following rules to drive:

1. Cars will move forward by their speed variable at each step, at the end of their step
2. If they are going less than their max speed, they will increase their speed by one

3. If there is a car in front of them prevents them from going their full speed, they will go as far as they can in that lane.
4. If a car is crossing the start of an intersection in a step and the light at that intersection is red, the car will stop at the intersection

The input of cars into the system requires that the first 15 feet in the simulation is clear according to the following:

1. Randomly choose one of the two lanes
2. If the lane is clear, add the car there
3. If not, check if the other lane has space
4. If neither are free, no car is added at that time

Activity Scanning Model

My model will use an event oriented implementation approach with an activity scanning model world view. With this approach, the simulation concentrates on the activities and especially the conditions that allow an activity to begin. At each clock advance, the conditions for each activity are checked and only if the conditions are true, will the corresponding next even occur. As the model is event driven, the simulation time will be continuous and state variables will be updated when an event has occurred. Events occur at irregular points in simulation time and simulation time advances from one event to the next. I will create a simulation application which contains the state variables, code modeling system behavior, and I/O and user interface software. This application will make calls to the simulation engine, which is calls to schedule events, another file in which the event list is managed and the management advances with simulation time. This file will call the simulation application as an event handler. For our random number generator we have all agreed upon random.Exponential. My model will share the same general assumptions as the event oriented queuing model, but whereas that model has B events only, mine will also handle C-events (condition types) and thus have a queue (FEL) for this too.

1. The C-condition is based on the event that a car has arrived at the traffic light - thus everytime the car has reached an intersection (except the 13th, because there is no traffic light at that intersection. When a B-type event reaches a light, it schedules a C-type event that waits for the light to be green when the light is green, all cars in the FEL of the C-type will go. The model operates by an event oriented queuing model that is based on events that go from intersection to intersection. The condition is, if the light is green, the car will move forward onto the next event by being added into the FEL for the B-events (so that car will be added to the FEL priority queue of going through the road aka the crossing event). Otherwise, the car must stay there till the light turns green, and if more cars reach this place and the light is still not green, then they are all waiting to move onto the next event on this que (they're added to the C-queue FEL until the light turns green).

Currently I am using anaconda environment and developing on python 3.6, the developing work can be found on the team's github.

Input Analysis:

Assumptions:

General Assumptions:

- Every car has same speed - calculated as 29 ft/s. We analyzed the NGSIM data and found the car that travels the entirety of the corridor the fastest and we found that to be 68 seconds. We divided the span of the corridor, 1938 feet, as summed from the span of individual sections and intersections which is detailed in the TrajectoryDataDescription.pdf that is provided with the data, with that rate of 68 seconds, to calculate the speed that we used for all cars. In the table to the left, from the provided NGSIM documentation, we summed the A1 length of all intersections (10th street intersection - 14th street intersection) and the length of sections 2-5 (between 10th and 11th, between 11th and 12th, between 12th and 13th, and between 13th and 14th)

Table 3-2: Intersection Lengths

Dataset	Intersection	Length, ft
A1	1	99.645
	2	129.795
	3	73.815
	4	66.979
	5	117.411
A2	1	99.732
	2	129.875
	3	73.513
	4	66.602
	5	121.319

Table 3-3: Section Lengths

Dataset	Section	Length, ft
A1	1	127.391
	2	441.437
	3	412.070
	4	353.727
	5	343.922
	6	11.730
A2	1	143.619
	2	417.976
	3	412.172
	4	351.511
	5	344.427
	6	28.636

- Cars are added to the world views using random.expovariate(1/mean) for generating interarrival times (in seconds) based on mean interarrival times for heavy, medium, and low traffic, which is 3, 7, and 12, respectively

Event Oriented Queueing model & Activity Scanning Model Assumptions:

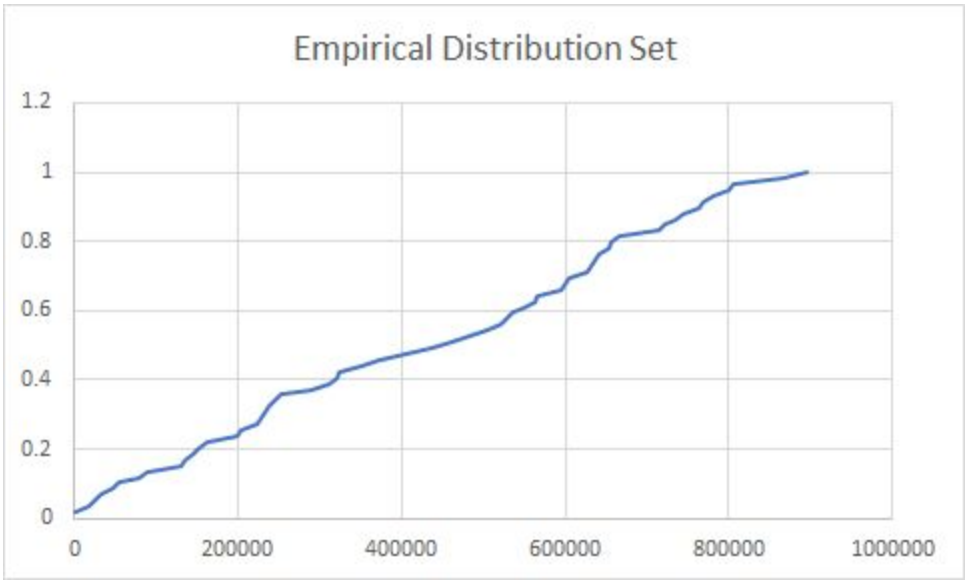
- Starts by entering 10th street intersection, if light is green
- Ist = intersection signal rotation times = green time + yellow time + red time
- Cars will arrive to a green signal if the global time mod the ist is less than the duration of green and yellow
- Otherwise cars will arrive during a red light and will have to wait until the red light is finished. The remaining duration of the red light can be calculated as the overall duration of the red light minus the global time mod the ist minus the duration of the green and yellow lights.

- The given speed of the car and length of the specific intersections and sections can be used to calculate how long it would take to cross that specific intersection or section. This will be used to calculate when future events should be scheduled
- If there is more than one car waiting for the green light at any intersection, then the crossing event will be scheduled rest_of_red + intersection_crosstime/2 time away to model that a waiting car cannot start crossing until the previous car is halfway through the intersection.
- The given left turn times from either NB or SB signals at each intersection shouldn't affect the given TR signal times for Northbound because the data there is not consistent with each other anyways. For example, if there is green for northbound, that is enough for cars to move through, independent of the left turn signal for either NB or SB.
- There is no signal light at the 13th street intersection so past 12th street intersection to beginning of 14th street are interpreted and combined to be modeled as one large section. Section 4 + intersection 4 + section 5 = one large section.

Cellular Automata Model Assumptions:

- All cars are 15 feet long
- Cars can stop directly at the back of other cars at intersections but will wait for a 5 ft gap between the car in front of them before they start to move
- Cars accelerate at $2ft/s^2$ until they reach their max speed of 28ft/s (since the model runs in half step increments, the speed is from two time steps and must be an integer for cellular automata

We used the NGSIM data set and took all instances of the vehicles that travelled all the way through the corridor, starting at the 10th street intersection and exiting at the 14th street intersection. We took all of the odd entries from this to use for the empirical distribution and left the even entries for validation.



From this graph, we can see that the cdf is relatively linear from the start to end time which means that the cars are arriving at approximately even intervals. From this, we know that we should use a mean arrival time and a distribution around that mean. We chose to use exponential distribution for a few reasons. First, a negative result would not make sense in any of the models, and exponential distributions never result in a negative number. Next, since this is simulating traffic, it is likely that the less traffic there is, the greater the variation between arrival times will be. With an exponential distribution, the standard deviation increases as the mean increases. To get these values, we used python's random.expovariate as described above.

Simulation Model:

The entire project can be found at https://github.gatech.edu/wvaughn9/PeachTreeSims. There are instructions in the readme to run the different models.

Verification & Validation:

Verification:

Event-Oriented Model: I verified my event-oriented model using a few different approaches. First I debugged to make sure that all the events were being popped in timestamp order from the FEL Queue. Then, I kept running counts of how many cars were waiting at the red signal of each specific intersection (which is calculated in the arrival event) and made sure that matched the number of waiting cars that traversed the intersection in the crossing event. I also kept count of the number of arrivals, crossings and departures that happened for each intersection and made sure that all three numbers matched per intersection, i.e. everything went through. I also made sure that the right number of cars were being generated by printing out debugging statements to track the overall process.

Activity Scanning Model: I verified my activity scanning model with several different methods. First I tested that the cars will be kept count of in the FEL of the C events list while they were "stuck" because of the light being red, so that I could be sure that they were in fact being pushed and popped into my priority queue data structure. Then, I also checked that even if the light is green, the cars will go into the FEL list anyway. I also checked the count for the number of arrivals, crossings, and departures occuring at each intersection. By keeping a count, I'll be able to make sure that the software terminated when it is supposed to.

Cellular Automata: I verified my simulation in a multiple ways. First, I tested that cars will not be able to exit the simulation if all of the lights are constantly red. Then, I checked that the cars should flow out of the system in the order they arrived when the lights are all green. Then I checked that cars stop at each of the red signals and do not move again until the light is green. Lastly, I tested that no cars can overlap within the simulation, including when placed onto the world. After completing these tests successfully, I moved to validation.

Validation:

The validation set that we used had a mean travel time of 135629ms with a confidence interval (130699, 140559)

Event-Oriented Model: I found that my simulation with various traffic levels has these average travel times:

Travel Level (Mean inter-arrival time in s)	Average Travel Time (in ms)
Heavy Traffic (3 s)	123404 ms
Medium Traffic (7 s)	121585 ms
Light Traffic (12 s)	120614 ms

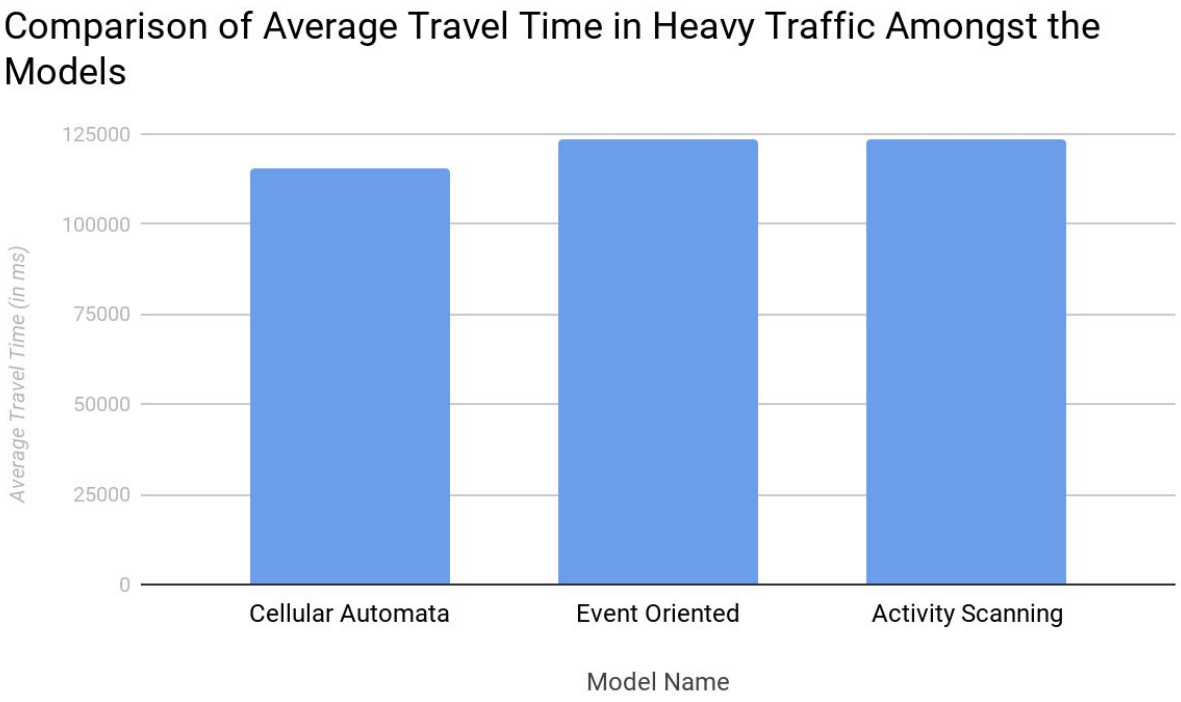
In all traffic levels, my cars took less time than the validation set. I attribute this to the oversimplification of my model in terms of not accounting for the physical systems events of lanes being full and cars having to wait at previous intersections regardless of the signal light color. Furthermore, my assumptions about the Northbound TR signals being independent of LT signals of both northbound and southbound at each intersection does limit other potential traffic-causing factors, which would explain why my cars are moving generally faster than the validation set.

Activity Scanning Model:

I found that my simulation with heavy traffic (3s) also had a very similar mean travel time of 123407 ms, with medium traffic (7s) 121585 ms, and for light traffic (12s) 120620 ms. This was expected because the activity scanning model is based off of the event oriented model, except it takes into considerations a C-event FEL to hold events (cars) that are waiting for light signal change, but this should not impact results much if at all.

Cellular Automata: I found that my simulation with heavy traffic had a mean travel time of 115496ms and a confidence interval of (113944, 117048). I expect the times to be lower than the validation set since my model starts at the exit of the 10th street intersection.

Comparison of simulators:



As can be seen in the above graph, Event Oriented and Activity Scanning performed very similar as expected because the two models are built very similarly. The minute differences can be attributed to the small conceptual differences in their architecture. Both models have understandably higher travel times than the Cellular Automata model because the Cellular Automata model starts at the end of the 10th street. All three models follow similar patterns for the other traffic levels as well and for the same reasons. These can be seen in the results section.

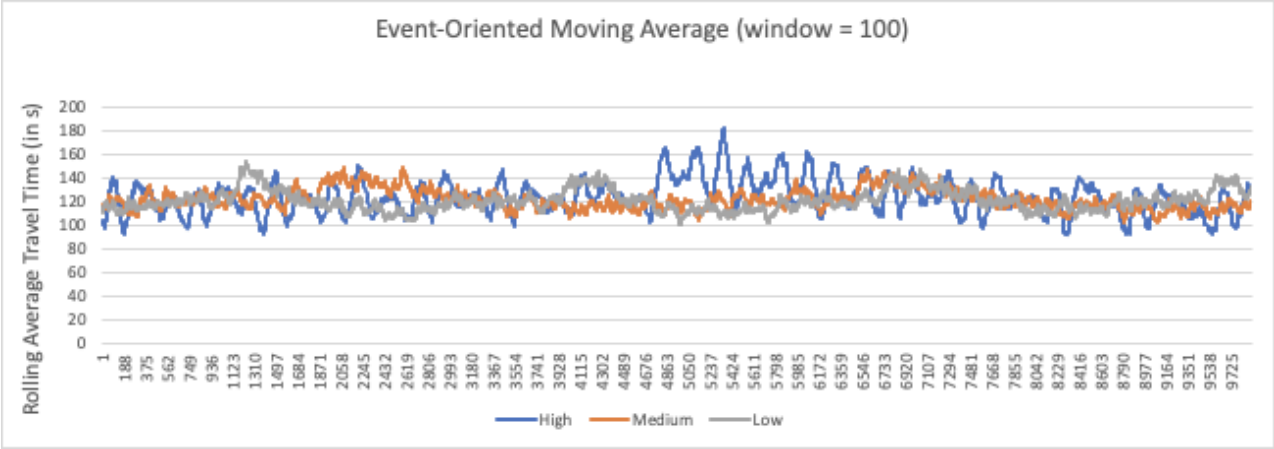
Design of Experiments and Output Analysis

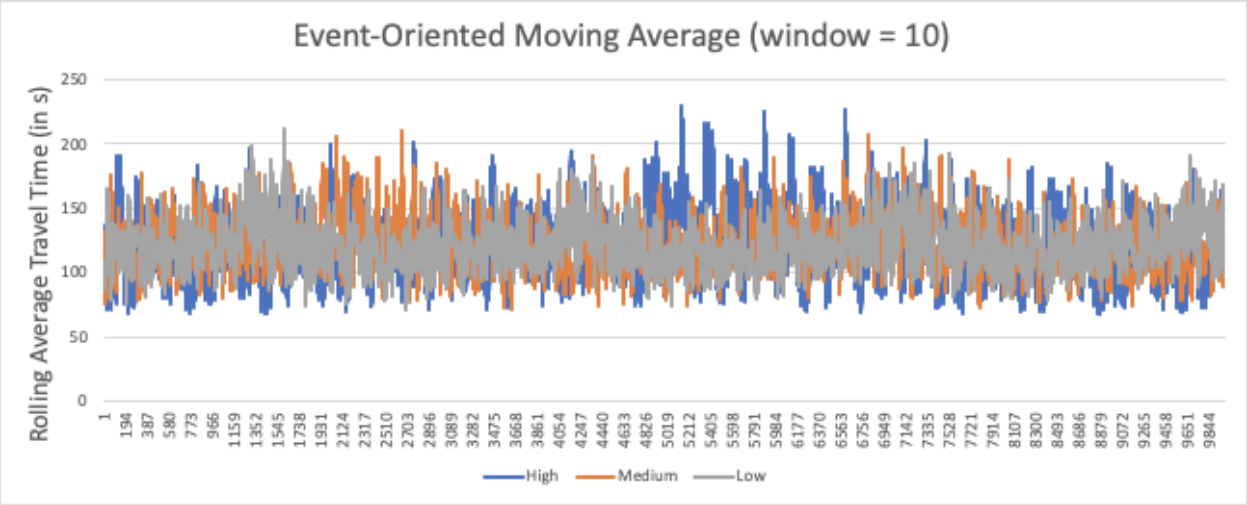
Experiment: Our goal is to test the average travel time, through the entire corridor, for various levels of traffic modeled by three different mean arrival times: heavy traffic (3 s), medium traffic

(7 s), and light traffic (12s). We hypothesize that higher levels of traffic will cause longer travel times. We will test this using each model and running each model for 10 simulations with a 1000 cars in each simulation. We will then use excel to calculate a 90% confidence interval for the travel times of each level of traffic, using *t*-distribution. We then calculated moving averages to see if there was a warm-up period.

Event-Oriented Model:

Travel Level (Mean inter-arrival time in s)	Average Travel Time (in ms)	Standard Deviation (in ms)	90% Confidence Interval
Heavy Traffic (3 s)	123404 ms	32471.6 ms	(122869.773, 123938.094)
Medium Traffic (7 s)	121585 ms	31387.7 ms	(121068.233, 122100.891)
Light Traffic (12 s)	120614 ms	31382.9 ms	(120097.551, 121130.052)



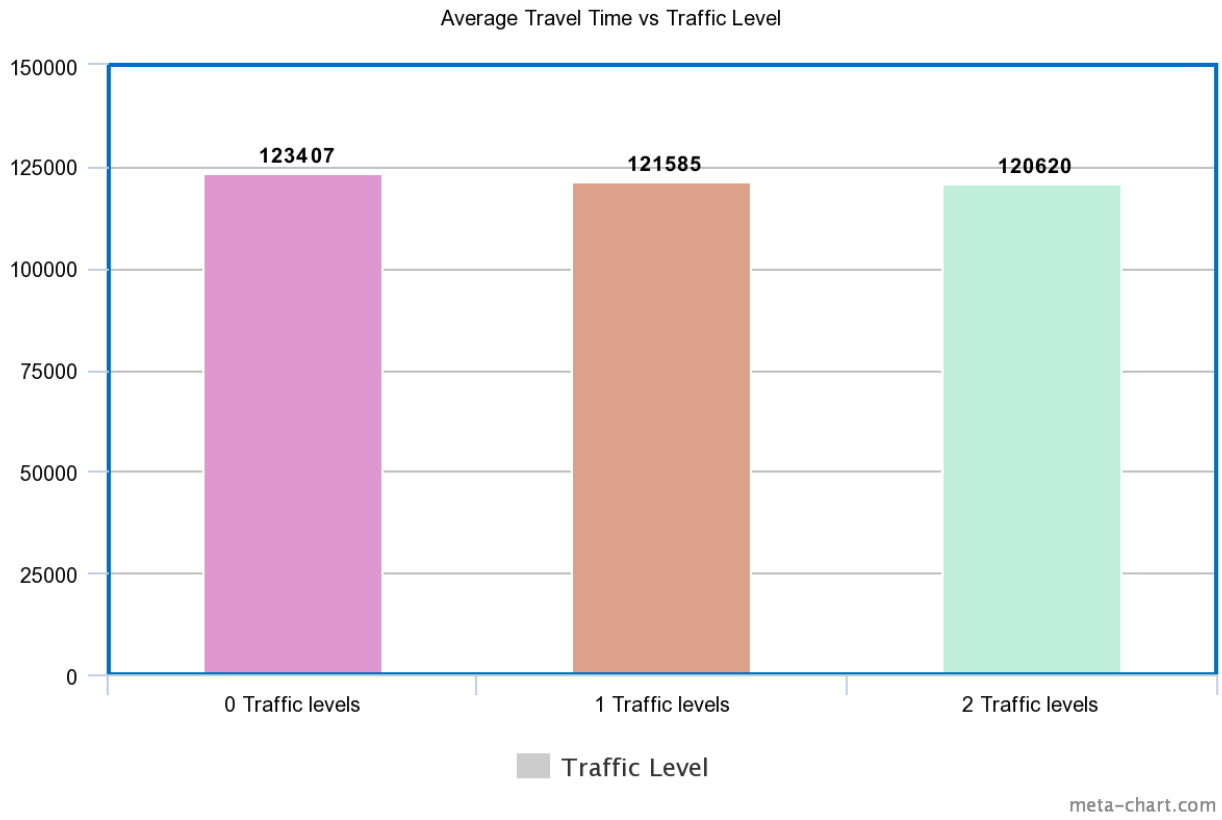


I calculated my moving average travel time for all 10,000 cars together and with both window = 10 and window = 100 and the data is shown above. In both graphs, there is no clear warm up period for any traffic level. This makes sense given the bounds of my simulation. As I have no limitations on the number of vehicles that can be in a given section and trigger events solely based on traffic signal cycle, all the cars can move together right away.

Activity Scanning Model:

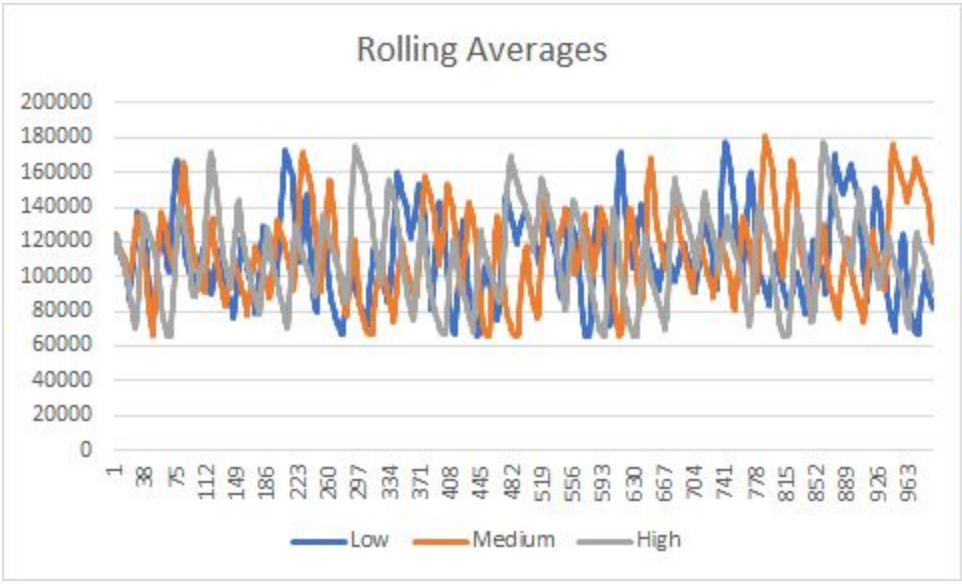
Travel Level (Mean inter-arrival time in s)	Average Travel Time (in ms)	Standard Deviation (in ms)	90% Confidence Interval
Heavy Traffic (3 s)	123407 ms	32472.8 ms	(122870.557, 124133.031)
Medium Traffic (7 s)	121585 ms	31388.5 ms	(121070.102, 122103.708)
Light Traffic (12 s)	120620 ms	31383.7 ms	(120100.163, 121672.013)

In my analysis of the model, I graphed the average times for high, medium, and low traffic levels into a bar graph so it could be easily compared to the other models. You can see from this graph that traffic does not affect the average time between medium (traffic level 1) and low (traffic level 2) very much, although there is a much larger difference between the aforementioned two and the high level of traffic average time.



Cellular Automata:

Before running tests on averages and confidence intervals, I wanted to look at warm up periods. I found the following by using a rolling average of size 10 for the each traffic level. The results can be seen in the graph below.



As it can be seen from the graph above, there is not clear warm up period for any traffic level. This makes sense given the bounds of this simulation. In the world view, there are three main

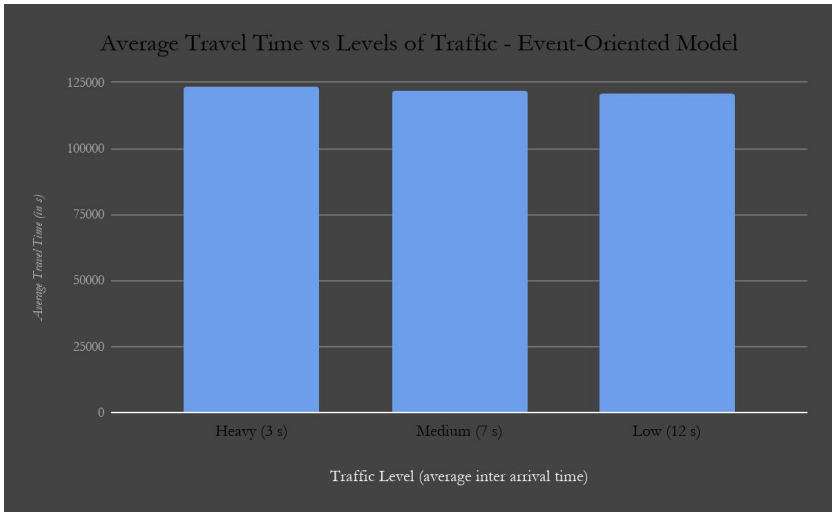
segments; between 10th and 11th, between 11th and 12th, and between 12th and 14th. The first two segments are approximately the same size, so the same amount of cars can fit in each. The only main factor controlling the output is the light at 14th which is why you can see such a big variation in waiting times. Most cars can get through each segment in one light cycle, and then in the last segment, they either have to wait awhile, or hardly no time. This is why the average time changes so much above.
In analysing the output of this model, I ran each of the traffic levels 10 times with 1000 cars in each iteration and found the following times for each low, medium, and high traffic.

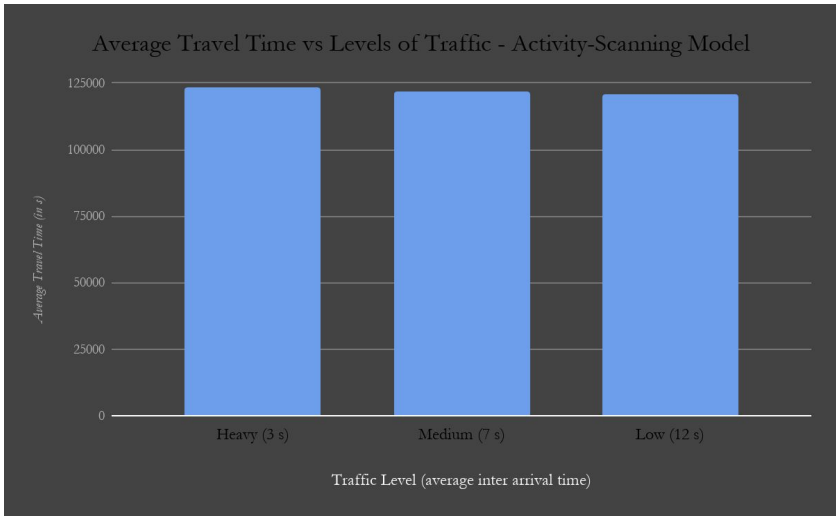
Travel Level (Mean inter-arrival time in s)	Average Travel Time (in ms)	Standard Deviation (in ms)	90% Confidence Interval
Heavy Traffic (3 s)	115496.4 ms	29834.9 ms	(112389.7, 117048.3)
Medium Traffic (7 s)	113990.5 ms	30776.2 ms	(112389.7, 115591.3)
Light Traffic (12 s)	111882.6 ms	30775.9 ms	(112137, 115338)

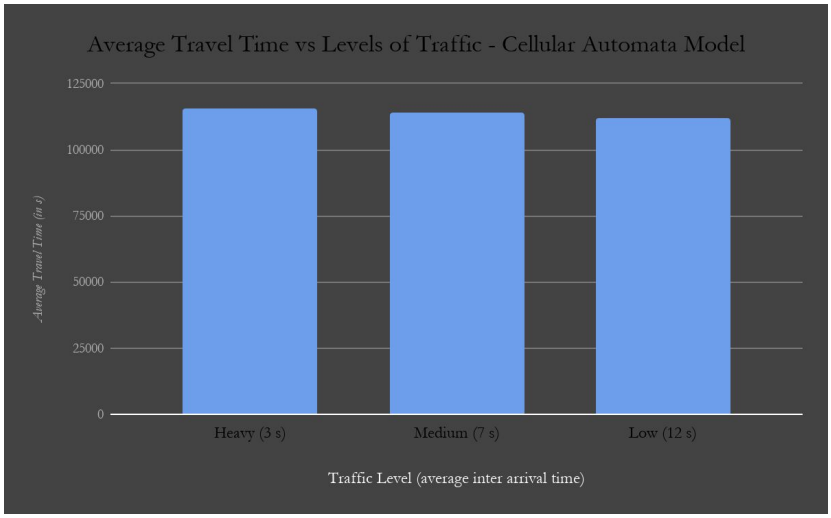
From this chart, you can see there is a slight difference between low medium and high, but it is a difference of just over 4.5 seconds between high and low traffic. The confidence interval of medium traffic overlaps with both the low and high traffic confidence intervals

Reporting Results

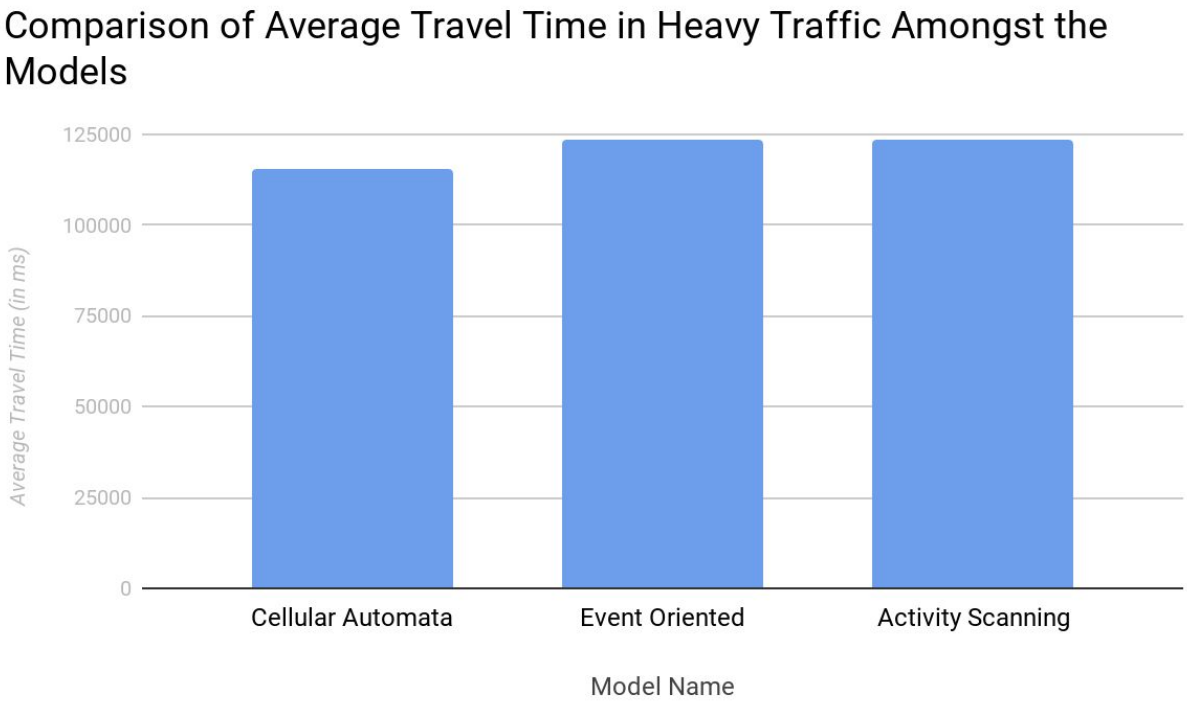
Our main goal is to analyze the relation between average travel time and levels of traffic using three models. The next three graphs will depict how average travel time is affected by traffic levels in each model.

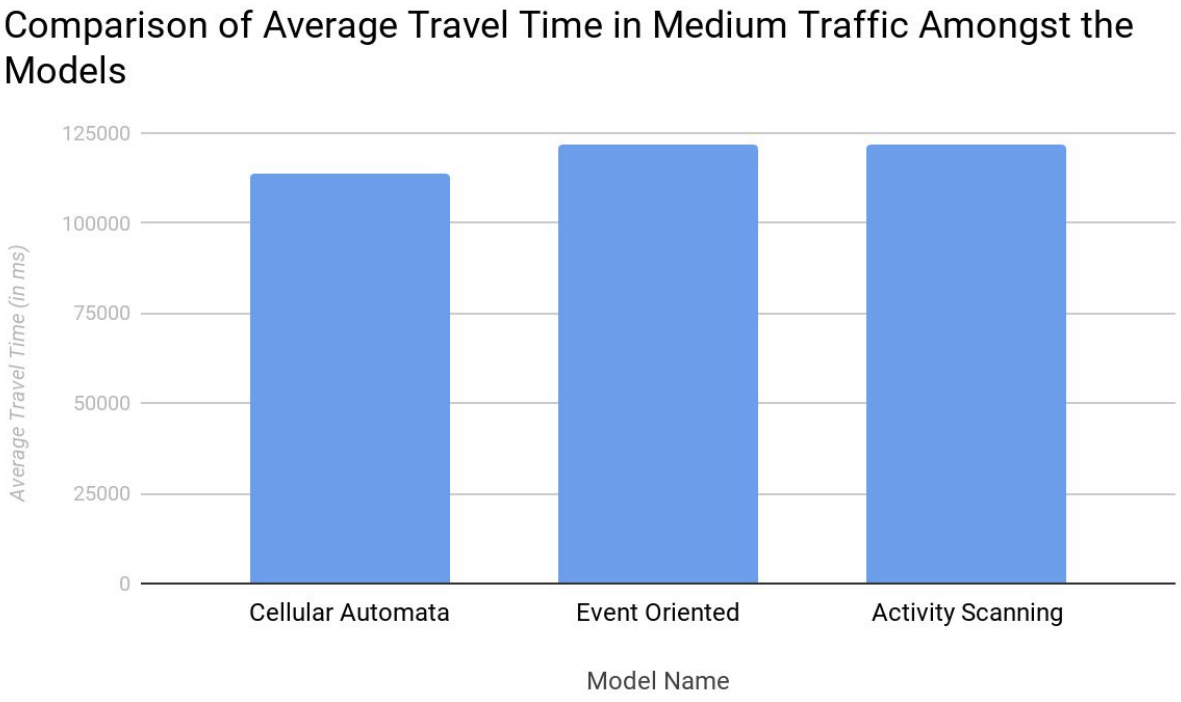

Average Travel Time vs Levels of Traffic - Event-Oriented Model
125000
100000
75000
50000
25000
0
Average Travel Time (in s)
Heavy (3 s)
Medium (7 s)
Low (12 s)
Traffic Level (average inter arrival time)


Average Travel Time vs Levels of Traffic - Activity-Scanning Model
125000
100000
75000
50000
25000
0
Average Travel Time (in s)
Heavy (3 s)
Medium (7 s)
Low (12 s)
Traffic Level (average inter arrival time)

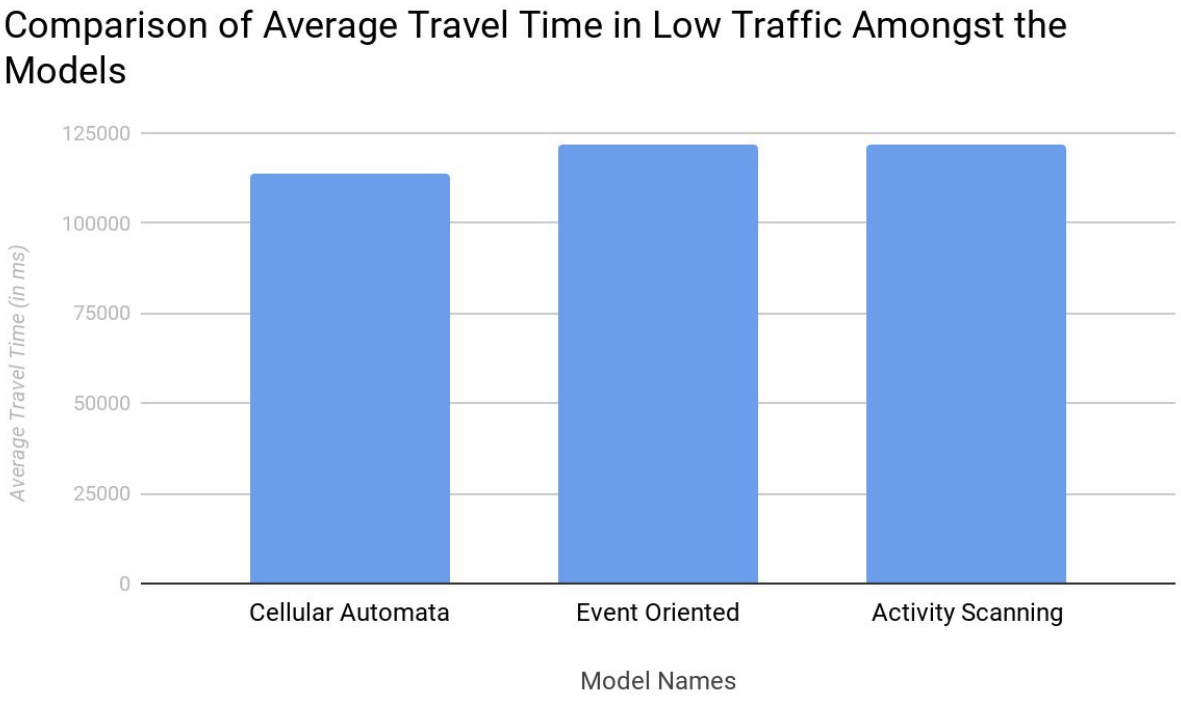

Average Travel Time vs Levels of Traffic - Cellular Automata Model
125000
100000
75000
50000
25000
0
Average Travel Time (in s)
Heavy (3 s)
Medium (7 s)
Low (12 s)
Traffic Level (average inter arrival time)

All three models follow the same trend of having the lowest average travel time for low traffic and highest average travel time for heavy traffic. Put together for each traffic level, the average travel times look like this:





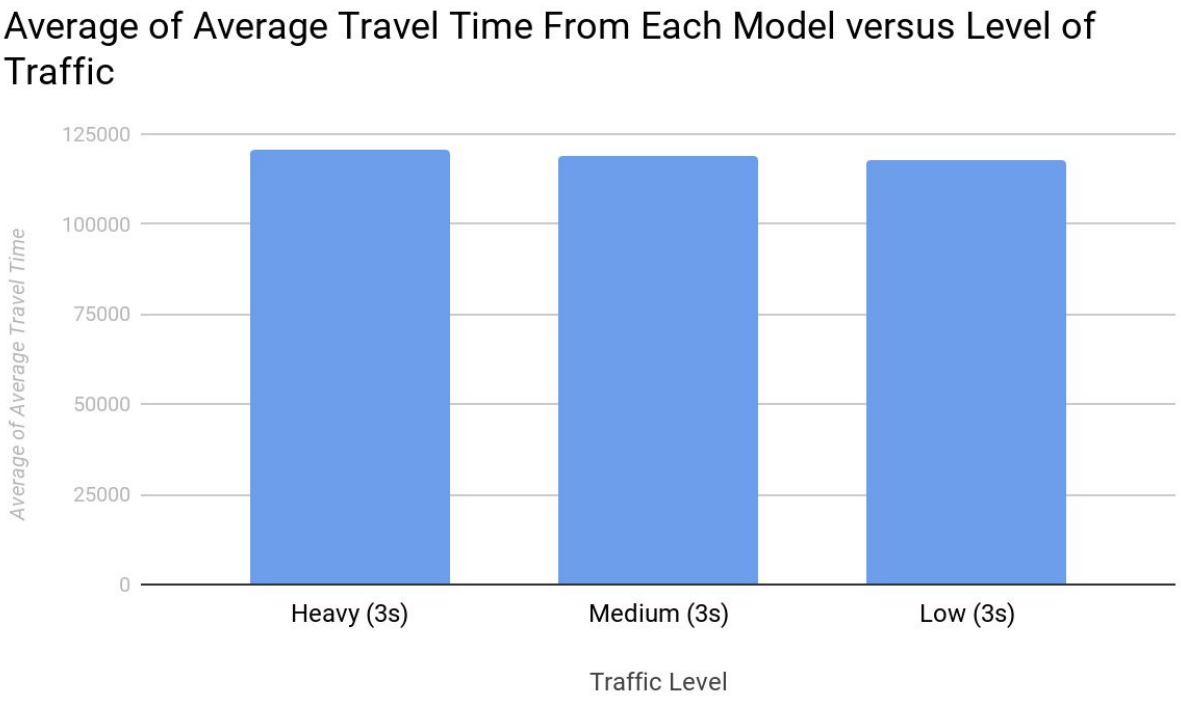
All three



All three graphs look nearly identical because each model followed the same pattern for each traffic level. From this, as we figured based on our assumptions and simplifications, we find that

the event oriented and activity scanning would be about the same time, and that both would take longer than the cellular automata model. This is once again because the cellular automata model adds car to the world view after the 10th street intersection while the other two might have to wait for the 10th street light to turn red.

For the final analysis, we decided to average the average travel time from each model and analyze it against each level of traffic.



By averaging the output from different models together, we have taken different approaches to understand the same phenomena: as we hypothesized, lighter traffic causes a faster travel time and heavier traffic causes a slower faster time. Amongst all three models, there is only small variations from each traffic level because of the simplifications and assumptions.