Simulating Traffic on Peachtree St NE

CX 4230 Project 2

Team 62

https://github.gatech.edu/bglowniak3/CX-4230-Project-2

Brian Glowniak and Joel Katz

4/23/2019

Table of Contents

1 - Problem Statement

For this simulation study, we will be modeling the flow of traffic through Peachtree St NE from 10th to 14th street, a corridor that runs through the midtown Atlanta area.

The end goal for our simulation study is to determine the distribution of times it takes a vehicle to fully traverse this section of roadway, given different levels of traffic (e.g. low-traffic, rush hour, etc.). Modifying the level of traffic will involve creating different scenarios of vehicle arrival times (e.g. heavy traffic would mean lots of vehicles entering the roadway).

We want to maintain as much accuracy as possible in other areas of the simulation (specifically, using accurate signal timings), but we may conduct further experiments on these parameters if we have time.

For the purpose of this project, we built two separate simulators. The first, designed by Brian Glowniak, follows the event oriented world view, while the other, built by Joel Katz, is designed as a cellular automata. Each simulator, while built separately, rely on the same input distributions.

This report consists of five major sections. First, the **Conceptual Model** section outlines the world views, entities, behaviors, and simplifications of our traffic models. Next, **Input Analysis** dives into how we mapped the Peachtree St corridor to a simulation representation as well as how we generated empirical distributions for various attributes from the provided NGSIM traffic data sets. The **Simulation Model** section then explores the specific implementation details of our models in Python 3. After that, in **Verification and Validation**, we describe how we verified our code and validated the results of our models compared to the real world data. Lastly, in **Output Analysis**, we compare the results generated from both simulators, build confidence intervals for the outputted statistics, and draw some final conclusions.

Our full codebase can be found on Github at this link. It will also be included again at relevant points in this report: https://github.gatech.edu/bglowniak3/CX-4230-Project-2.

2 - Conceptual Model

For this project, we created two types of models: event-oriented and cellular automata. In this section, we describe the conceptual models devised for each world view, including entities, attributes, events, and simplifications.

2.1 Event-Oriented

The event-oriented model is modeled as a queueing network of intersections. Each intersection/traffic light is a server with a queue for each lane that is first come first serve. The simulation progresses as vehicles travel from intersection to intersection.

This conceptual model defines two major entities:

- **Vehicle** - this is the main entity that traverses the simulation. In the queueing network paradigm, these are the **consumers** in the model. Their main behaviors include: arriving at intersections, queueing at intersections, and traversing between intersections. A vehicle has an ID, a simulation entrance time, a simulation exit time, an entrance intersection, an exit intersection, a direction, a lane, and a velocity. We define two directions: North and East/West. A vehicle traveling north is currently traversing the corridor, while a vehicle traveling East/West is entering the corridor from an intersection along the route. There are only two lanes (left and right).
- **Intersection** - In the queueing network paradigm, these are the **servers** in the model. They act as nodes in the network along the corridor, and each has a queue corresponding to each lane and direction combination (four queues in total). Each intersection also has a stoplight state (green or red), a green duration, a red duration, the distance to the next intersection, and the length of the intersection.

The simulation functions through the execution of five main events:

- **Simulation Arrival** - a vehicle arrives at the corridor. This event will then schedule an Intersection Arrival event based on which intersection the vehicle specified as its entrance point. The event will then spawn a new vehicle (more details in Section 3 - Input Analysis) and schedule a new Simulation Arrival event based on that vehicle's generated interarrival time.
- **Simulation Exit** - a vehicle exits the corridor. Once a vehicle reaches the intersection that it designated as its exit point, this event logs some information from the vehicle's traversal of the corridor such as total travel time. This event does not schedule any other events.

- **Intersection Arrival** - a vehicle arrives at an intersection. If the vehicle is traveling north and the light is green, or if the vehicle is traveling E/W and the light is red (therefore green in that direction), the simulation will schedule an intersection departure event and the vehicle will traverse the intersection. If the vehicle is traveling E/W and the light is green, or if the vehicle is traveling north and the light is red, the vehicle will queue in the appropriate queue according to what lane it is in and what direction it is traveling. Any vehicles arriving at the 13th St intersection will be able to depart as soon as they can, as there is no signal there. When a vehicle queues, if the size of the other queue is less than 75% the size of the queue it is about to join, it will change lanes. This is to simulate cars changing lanes to avoid traffic, and the 75% represents the largest size where a difference may actually be noticeable and therefore worth it to move.
- **Stoplight Change -** the stop light at an intersection changes color. When the light turns green, the event will check for vehicles queueing in the northbound direction, and if the light turns red, the event will check for vehicles queueing in the E/W direction to allow them to enter the corridor. In either case, the waiting vehicles will be dequeued and the event will schedule an intersection departure for each. It is important to note that vehicles do not instantaneously traverse the intersection. According to page 11 of the TrajectoryDataDescription.pdf in the NGSIM dataset, this delay is around 2.5 seconds. Based on this fact, the first vehicle in line will wait between two and three seconds to depart, and additional vehicles will wait up to one second after the vehicle ahead of it has left. This is meant to realistically simulate how a line of vehicles takes a bit of time to get moving. Lastly, at the end of execution, this event also schedules the next stoplight change for that intersection based on signal timings.
- **Intersection Departure** - in the context of a queuing network model, this represents an entity utilizing a server. If the current intersection is the exit point of the vehicle, then the event will schedule a simulation exit event and allow the vehicle to leave the corridor. Next, because vehicles experience delays in movement when a stop light turns green, this departure event checks to see if the light has turned red again. If it has, then the vehicle cannot depart the intersection, and is subsequently requeued. If a vehicle successfully makes it through, regardless of what direction it entered from, it will begin to travel to the next intersection. As a result, this event also schedules an arrival at the next intersection for the departing vehicle based on the velocity of the vehicle and the distance to the next intersection.

The conceptual model doesn't consider anything that may occur between intersections (lane changes, collisions, road curvature, etc.). In terms of events, once a vehicle passes through an intersection, it is not considered until it arrives and queues at the next one.
Assumptions made (some of these also apply to the CA model and are reiterated in that section):

- The DES sim does not consider the size or length of any vehicles. This is because any delays by longer vehicles are assumed to be negligible.
- The simulation consists of two lanes for both the north and E/W directions. We do not account for left turn lanes or signals as we felt that the effect of this is negligible . Vehicles do not change lanes with the exception of the case described in the Intersection Arrival event.
- Vehicles can only enter and exit at the major intersections along the route. To account for side streets and driveways, we map each to the nearest intersection in the northbound direction. This is outlined in more detail in Section 3.1 of Input Analysis. The reason for this is after exploring the data set, we found that the number of cars entering from those streets was fairly negligible.
- Vehicles never accelerate or decelerate (speed is binary - either the max speed or 0). A vehicle is assigned a velocity based on an empirical distribution when it is first spawned into the simulation.
- Vehicles cannot collide, and the speed of a vehicle is not affected by any other vehicles. If a faster vehicle is caught behind a slower one, the simulation will handle this by having that vehicle reach the next queue first. If there is a sizeable difference between vehicle speeds, this represents how a faster vehicle will pass a slower one. If the difference is small, the effect it has on travel time is negligible.
- Traffic beyond 14th Street is not considered. Once a vehicle passes 14th, it is effectively removed from the simulation. As such, we do not consider vehicles that enter at the 14th St intersection as they will immediately leave the simulation.
- Yellow lights are not considered. To account for this, half of the yellow duration is added to green, and half is added to red. This represents how some vehicles will continue while others will stop when a light turns yellow.

2.2 Cellular Automata

In this model, the road section is represented by cells, one cell for each section of the road in which one vehicle could fit. We define one cell as roughly 20 feet long and 10 feet wide to match an average car length and average lane width. A cell can contain one or zero vehicles.

A vehicle is an object whose motion is governed by the characteristics speed, max speed, and gap (space between this vehicle and the preceding vehicle). A vehicle's speed is a function of these variables, with these rules defined as follows. Note i is the vehicle, s(i) is its speed, m(i) is its max speed, g(i) is the gap, and random() is a random number on the interval (0,1).

If $s(i) < m(i)$, Then $s(i) := s(i) + 1$	E1
If $s(i) > g(i)$, Then $s(i) := g(i)$	E2

If random() < P, Then s(i) = s(i) - 1 E3

In equation E1, this rule accounts for linear speedup of the vehicle. Here we assume that a vehicle will speed up linearly. Equation E2 accounts for slowdown to avoid crashing into the vehicle ahead of it. This assumes that a car will only be willing to travel a speed for which in one second it will not reach the location of the next car, otherwise known as a one second following distance. E3 accounts for random slowdown, since if the random number falls below the threshold P, the vehicle slows down an additional unit. Here we make the simplification that drivers randomly slow down according to a Bernoulli distribution with probability P, which doesn't vary between vehicles.

The unit of speed is used to advance the vehicles in a timestep, more specifically, the vehicle is advanced one cell per unit speed. We define one cell update as happening once per second, a cell has length 20 feet (as defined earlier), and we give a vehicle a maximum speed of 5. Using these values, the vehicles can travel speeds in the range [n*20 feet/second] for n = [0,1,2,3,4,5]. Giving a maximum speed of 100 feet per second, or about 68 miles per hour. These numbers give a convenient method of defining real world values for the simulation parameters.

The model is that of a two-lane road, represented as a two dimensional grid of cells with width two. This model allows for lane changing and passing, further mimicking real traffic. Lane changing is governed by a set of three predicate equations shown in E4, E5, and E6. Note i is the vehicle, g(i) is the gap ahead of vehicle i, go(i) is the gap ahead of the vehicle if it were in the other lane (returns -1 if there is a vehicle next to i), gb(i) is the gap behind the vehicle were it in the other lane (returns -1 if there is a vehicle next to i), and random() is a random number from a uniform distribution on (0,1).

go(i) > g(i) E4
gb(i) > E5
random() < P E6

If all of these predicates evaluate to true in a given time step of the simulation, then the vehicle will change lanes. These equations represent whether the gap in the other lane is greater than the gap is in the current lane, meaning the driver has more space if they move over, whether there is at least two spaces upstream so the driver will be comfortable changing lanes without cutting another driver off, and whether a random sample from a bernoulli distribution with probability P evaluates to true. These together form a logical model for how a real vehicle might decide to change lanes.

Once vehicle decides to change lanes, it simply is moved horizontally. Real vehicles cannot move directly horizontally, but when this is combined within a single time step that also includes forward advancement based on the rules defined earlier, in one timestep the vehicle will change lanes and move forward, matching a logical model of vehicle behavior.

In a lane, there can be any number of stop lights. These stop lights are an object that have a location along the lane, a value of red or green, and a timing for how long to stay red and green. Using the NGSIM data of Peachtree St, specifically the seconds of time that each stop light remains red, yellow, or green, we gave the stop lights initial values that match the real world values, making the light change after n iterations, which we already defined as being exactly one second. For simplicity, the yellow light time was split between green and red since there is a threshold within the yellow light wherein vehicles will stop, which was assumed to be around the middle of the yellow light.

The random generation of inputs to this model will be further discussed in the Input Analysis section, but for the application to this model, when a random vehicle is generated, it is placed in a queue matching the input location. For example, if a car is set to originate from Eastbound on 12th street, it will be queued in the corresponding queue, and when the light changes to red for Northbound, cars will dequeue and enter the simulation. Therefore, cars are able to be scheduled even when they can't enter the two-lane model yet, allowing for a more accurate representation of inflow from side-streets.

The inflow of traffic from Peachtree street before 10th Street is implemented slightly differently. There is no queue for traffic entering here, because if cars inflow constantly here and are added to a longer and longer queue, one would expect that cars would back up further and further into the queue, which would become a representation of the road behind the queue, which would skew the entrance time into the section of road we are actually studying. Therefore, if a car cannot be added, it is simply discarded and assumed that the interarrival time would slow in this case.

Once traffic passes through the end of the lane (the 14th street stop light), the car is assumed to have a gap of 5 (maximum effective amount) and will exit the simulation on the subsequent timestep. Therefore, it is assumed that traffic following the 14th street stoplight is negligible.

Enumeration of simplifications:

- Only a single vehicle occupies exactly one cell of size 20 feet by 10 feet. A vehicle is never midway between cells.
- A car may travel 1 of six speeds [n*20 feet/second] for n = [0,1,2,3,4,5]

- A vehicle speeds up linearly, and will never travel further in one timestep than the distance between it and the preceding vehicle (minimum one-second following distance).
- Any vehicle slows down one unit of speed (20 feet/second) with probability P.
- The duration of a yellow light is treated as one-half green and one-half red.
- The arrival of vehicles along Peachtree Street behind 10th street are only counted if there is space available, and therefore if there is not space available, the interarrival time increases.
- Traffic following the 14th street intersection is negligible
- There are never any collisions.
- There are no turn lanes
- The curvature of the road and other obstacles have no effect on the simulation

3 - Input Analysis

All code, datasets, and files referenced in this section that are related to our input analysis processes are located in our Github repository in the */Input Analysis* folder:

https://github.gatech.edu/bglowniak3/CX-4230-Project-2/tree/master/Input%20Analysis

We performed all of our input analysis in a Jupyter notebook titled *Input Analysis.ipynb*. The functions and distributions were then added to a file called *simulation_input.py* that is in each simulator's directory. The simulation models call the functions from this file when generating vehicles.

3.1 Peachtree Corridor and NGSIM

To start our input analysis, we first explored the Peachtree St NE corridor to better understand how to map it to a conceptual model. Physical characteristics investigated included the number of signals and intersections, signal timings, and the distances of the various segments of the corridor.

For the purpose of our model, we have chosen to only explore the corridor in the northbound direction. The section of roadway we are analyzing is two lanes and passes through five major intersections (10th, 11th, 12th, 13th, and 14th St). 13th St is only a right turn and does not have a stop light, while all others are four ways with traffic signals.

To compute the distances between intersections and the length of each intersection, we utilized tables 3-2 and 3-3 included on page 7 of the Trajectory Data Description included with the NGSIM dataset. Each included two measurements (in feet) of these distances, so for our simulation we took the average of the two and converted it to miles.

To compute signal timings, we utilized the NGSIM signal timings spreadsheet (*signalTimings.xls)*. Due to the simplifications made in our conceptual models, we have only used Green, Yellow, and Red Northbound TR data. To account for yellow lights, we divided the yellow timing in two and added half to the green duration and half to the red duration. This accounts for how some vehicles may start and some vehicles may stop.

The roadway information collected is displayed in the table on the next page. These values are all currently included in the simulation models as state variables.

Table 3.1.1: Physical Characteristics of Peachtree St NE

Intersection	Length (mi)	Distance to Next (mi)	Green Duration (sec)	Red Duration (sec)
10th St	0.0189	0.0814	36.5	51.1
11th St	0.0246	0.0781	43.1	57
12th St	0.0140	0.0668	62.5	37.3
13th St	0.0126	0.0652	N/A	N/A
14th St	0.0226	N/A	36.2	47.7

Once we compiled this physical roadway information, we began to work on the NGSIM dataset to explore the behavior of vehicles entering the system. We first filtered all data cases by only those traveling in the northbound direction, then removed columns that we wouldn't be considering (such as Local_X, Veh_Len, etc.). Then, since there were a multitude of entries for each vehicle, we grouped by vehicle ID and created one entry per vehicle. This new entry included new values such as entrance time (the minimum of Epoch_ms), exit time (the maximum of Epoch_ms), as well as average and max velocity.

Next, to account for our simplification in the conceptual model that vehicles can only enter and exit at major intersections, we remapped all origin and destination zones to the nearest intersection in the northbound direction. The simulation mappings below will be referenced in 3.3.1 and 3.3.2 when generating the discrete distributions for entrance and exit points.

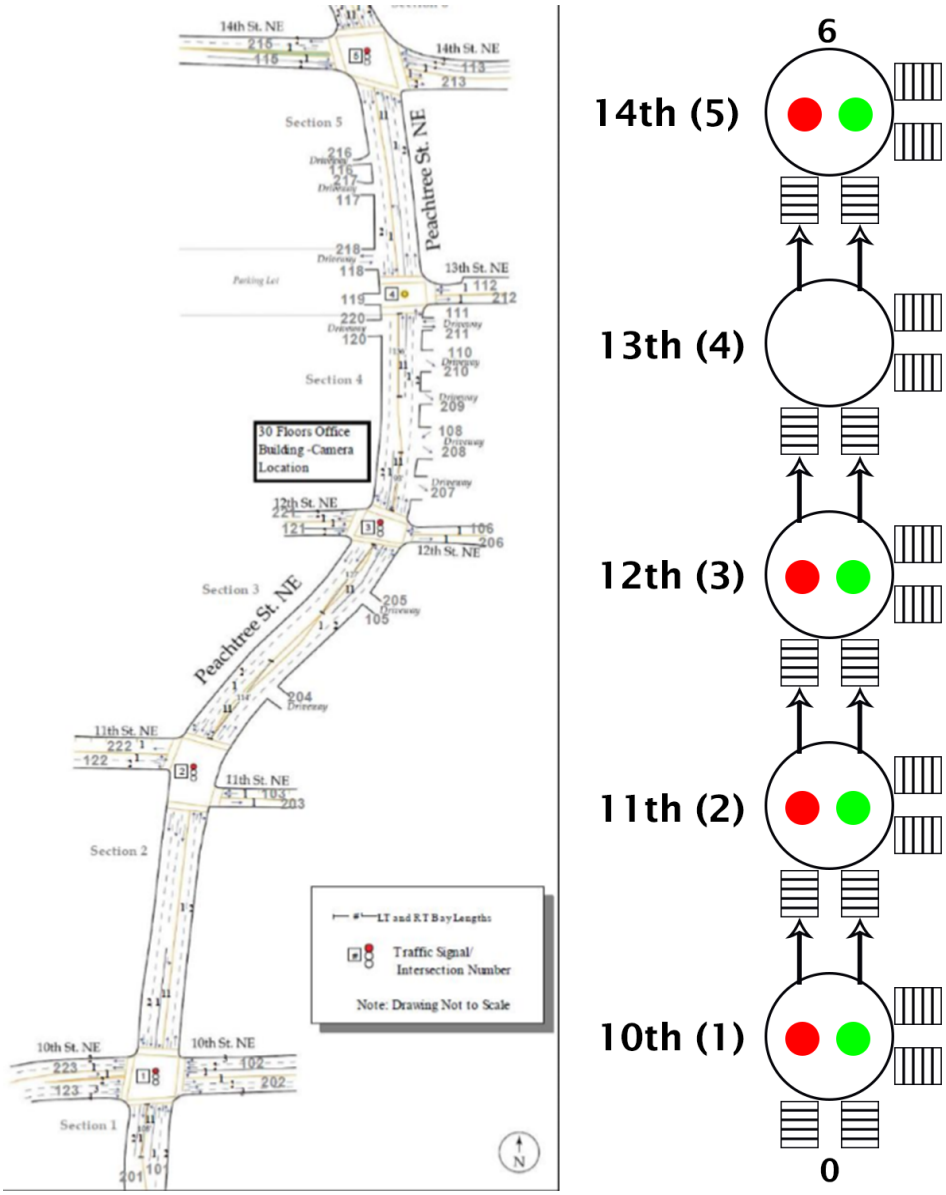
Table 3.1.2: Simulation Model Zone Mappings

Simulation Mapping	Origin Zones	Destination Zones
Before 10th St (0)	101	N/A
10th St (1)	102, 123	202, 223
11th St (2)	103, 122	203, 222
12th St (3)	105, 106, 121	204, 205, 206, 221
13th St (4)	108, 110, 111, 112, 119, 120	207, 208, 209, 210, 211, 212, 220
14th St (5)	113, 115, 116, 117, 118	213, 214, 215, 216, 217, 218
After 14th (6)	N/A	214

After mapping the data, we realized that some data cases were reported erroneously. There were a small handful of vehicles that were logged as entering at Origin Zone 114 and leaving at Destination Zone 214, which corresponded to driving south towards 14th St, making a U-Turn, and driving back north. Another handful were logged as entering at Origin Zone 101 and exiting at Destination Zone 201, which corresponded to the same movement just at 10th St. We removed these cases.

Lastly, to begin work on analyzing the data and generate distributions, we split the processed dataset into two parts: the model set (*Processed_Vehicle_Data_Model.csv*, length 157) and the validation set (*Processed_Vehicle_Data_Validation.csv,* length 156). We used the model set to generate our distributions, and we used the validation set to test our models against in Section 5.

Figure 3.1.1 (below): Peachtree St NE and its corresponding simulation representation



3.2 Continuous Distributions

3.2.1 Vehicle Interarrival Time

The most important part of our simulation is creating a way to simulate traffic flowing into the system. The best attribute to capture this was interarrival time, or the time between when two subsequent vehicles enter the system. We computed this attribute for each vehicle in the model set. The process of how we created this distribution is outlined as such:

First, we wanted to test that interarrival time was an independent stochastic process. To do so, we first generated a scatter plot of vehicle ID vs interarrival time.

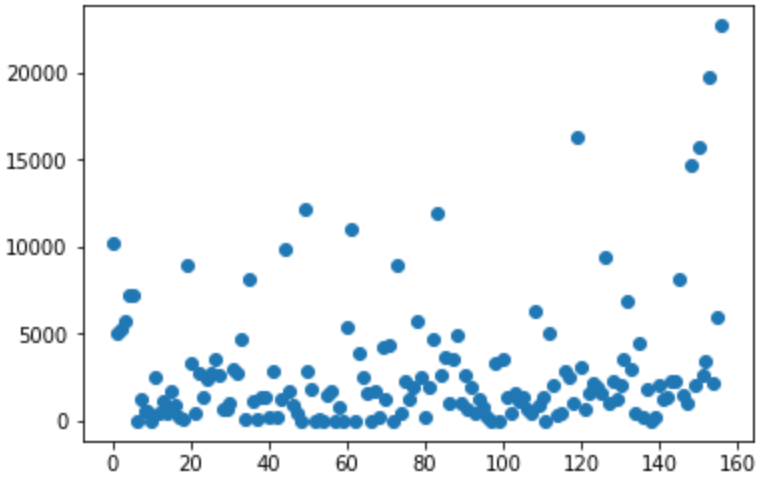


Figure 3.2.1: Plot of vehicle vs. interarrival time

It is clear from this scatter plot that there is no correlation between subsequent values of interarrival time. To further emphasize this, we also created an autocorrelation plot based on the following formula:

$$\widehat{p}(k) = \left(\sum_{i=1}^{n-k} (x_i - \bar{x}(n))(x_{i+k} - \bar{x}(n))\right) / ((n-k) * s^2(n))$$

While a scatter plot depicts correlation between subsequent data points, an autocorrelation plot determines the correlation between data points separated by a span of k samples. The result is depicted on the next page.

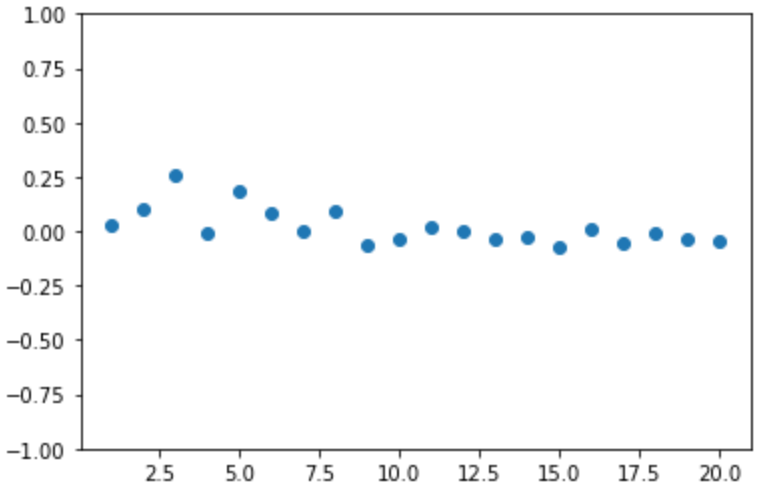


Figure 3.2.2: Autocorrelation plot for interarrival time

Since the majority of values plotted hover around zero, this indicates with certainty that this variable is independent. To begin computing the empirical distribution, we first need to generate a histogram of the data. We select the number of bins as the square root of the number of data cases in our model set: $\sqrt{157} \approx 13$. We want to ensure that the curve is neither too jagged or too smooth

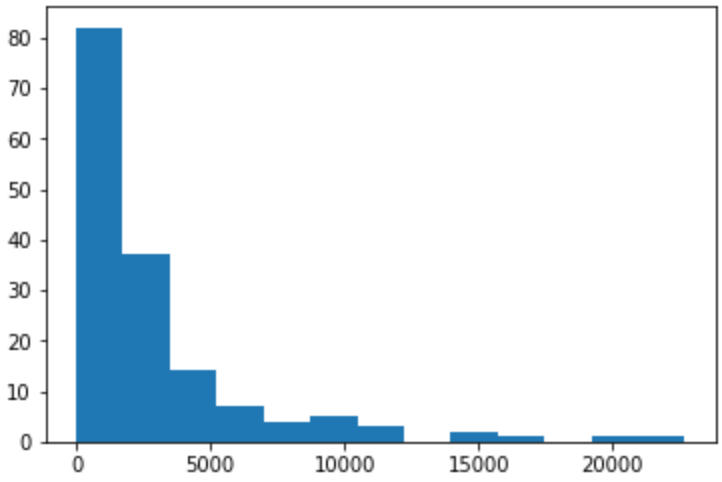
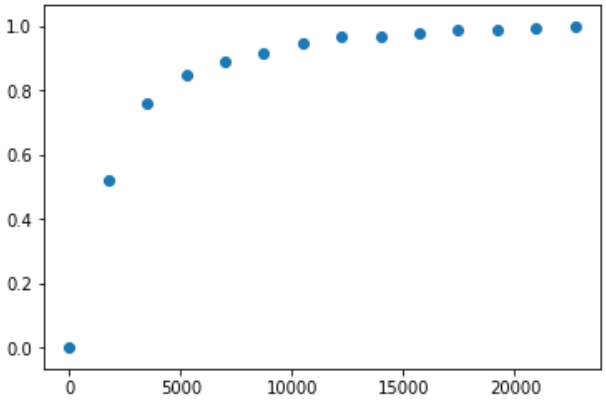


Figure 3.2.3: Histogram of interarrival with 13 bins

Below is the generated CDF (*Figure 3.2.4)* and empirical distribution. The CDF was generated by taking the right boundary of each bin for the interval values, and the number of samples in that bin over the total number of samples for the CDF value.



Class Interval Right Boundary (ms)	CDF F(x)
1746.1538	0.5223
3492.3077	0.7580
5238.4615	0.8471
6984.6153	0.8917
8730.7692	0.9172
10476.9231	0.9490
12223.0769	0.9682
13969.2308	0.9682
15715.3846	0.9810
17641.5385	0.9872
19207.6923	0.9873
20953.8462	0.9936
22700	1.0

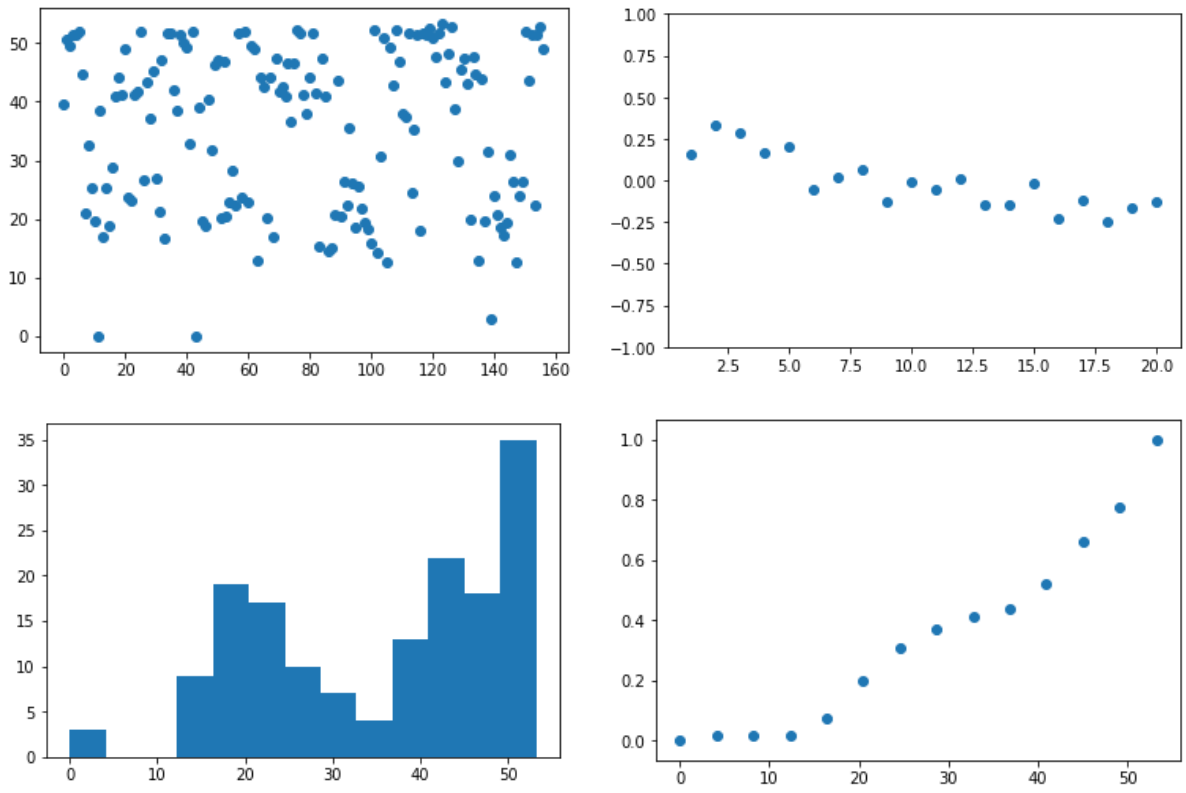
To calculate values from this distribution, we first generate a random number between (0, 1). From that number, we determine which class interval we are in (i.e. which two CDF values are we between?). Then, since this is a continuous distribution, we linearly interpolate between the two values to find an interarrival time. The resultant time is then converted to seconds for the simulation model to use.

To account for different levels of traffic, we added a parameter called Scale Factor that scales the interarrival time by a specific value. For default, medium traffic, the value is 1. For increased traffic, we decrease the factor to around 0,5, and to decrease traffic, we increase the factor to 1.5 - 2.0.

3.2.2 Vehicle Velocity

We also decided to generate an empirical distribution for vehicle velocity to more realistically mirror the real world. The process was identical to generating a distribution for interarrival time.

Figures 3.2.5 - 3.2.8 (clockwise): Scatter plot of vehicle vs velocity, autocorrelation plot, histogram, and generated CDF



The resultant distribution is shown below. Calculating values from it is the same process described in the interarrival section.

Class Interval Right Boundary (ft/s)	CDF F(x)
4.0954	0.01911
8.1908	0.01912
12.2862	0.01913
16.3815	0.0764

20.4769	0.1975
24.5723	0.3057
28.6677	0.3694
32.7631	0.4140
36.8585	0.4395
40.9538	0.5223
45.0492	0.6624
49.1446	0.7771
53.24	1.0

3.3 Discrete Distributions

3.3.1 Entrance Intersection

In order to effectively model vehicles entering at any point in the corridor, we generated a discrete distribution based on the NGSIM dataset. The histogram generated from the model set that we used to do so is below. The mappings are as described above in section 3.1.

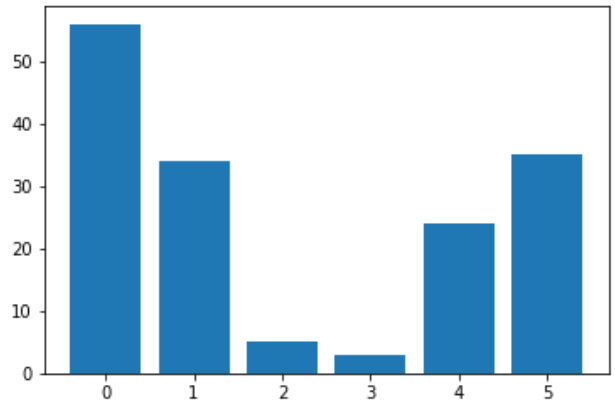


Figure 3.3.1: Histogram of Entrance Intersection

When generating the distribution, we excluded the bin at zone 5, as this represents vehicles entering at 14th St. For the purposes of our simulation, these vehicles immediately exit, so they are not considered.

Table 3.3.1: Discrete Distribution for Entrance Intersection (N = 122)

Zone	N(x)	PDF p(x)	CDF F(x)
0	56	0.4590	0.4590
1	34	0.2787	0.7377
2	5	0.0410	0.7787
3	3	0.0246	0.8033
4	24	0.1967	1.0

3.3.2 Exit Intersection

Similar to generating a distribution for entrance point, we also wanted to generate one for exit point. The process was identical, except in this case we used zones 1 through 6, as vehicles can exit at any point except for Zone 0 (before 10th St). Zone 6 represents exiting beyond 14th St.

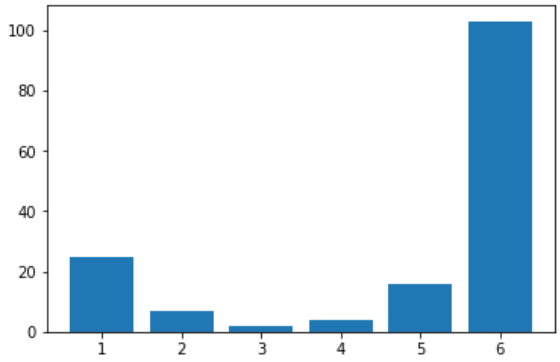


Figure 3.3.2: Histogram for Exit Intersection

Table 3.3.2: Discrete Distribution for Exit Intersection (N = 157)

Zone	N(x)	PDF p(x)	CDF F(x)
1	25	0.1592	0.1592
2	7	0.0446	0.2038
3	2	0.0127	0.2166
4	4	0.0255	0.2420
5	16	0.1019	0.3439
6	103	0.6561	1.0

4 - Simulation Model

As stated in the introduction, our simulation model code is on Github at the repo linked here:

https://github.gatech.edu/bglowniak3/CX-4230-Project-2

The */ca* directory contains the cellular automata simulation, while the */des* directory contains the discrete event-oriented simulation. Further information on the code structure and running the code can be found in the README.md file located in the repo. As mentioned previously, Joel developed the CA sim, while Brian developed the DES sim.

4.1 Libraries and Helper Files

For this project, our simulations were built in Python 3 with the help of a handful of Python libraries.

- queue - used to implement the FEL via PriorityQueue in the DES sim
- random - used to generate random variables via empirical distributions that were created in the input analysis phase
- enum - used to create an enumeration of intersections (since the set is discrete and finite)
- time - used to determine total runtime of a simulation run by getting the start and end times via time.time()
- ABC/abstractmethod - in the DES sim, events subclass an ABC (abstract base class) that the simulation engine provides as a scaffold of sorts.
- pandas - in the cellular automata as well as much of the analysis is done using this data manipulation library, which is capable of producing many of the SQL-like manipulations we perform on tables of data

5 - Verification and Validation

5.1 Model Verification

5.1.1 Event-Oriented Model

At the core of the DES simulation is an object called SimulationState that can accept user-inputted parameters to the simulation. This object is passed through each event execution as a global variable to manage simulation state and track statistics based on the inputted parameters. One specific parameter is called *debug_flag*, which when set to True, prints the result of every event computation along with the timestamp of execution to the console. Another parameter, *max_departures*, allows us to control how many vehicles leave the system before the simulation terminates. Lastly, we can also control and/or limit vehicle generation in the SimulationArrival event. Using these options, we can create traces of controlled simulation runs that we can view on the command line. Using these outputs, we can assess whether the event computations are what we expect them to be as outlined in the conceptual model. Below is an example command line output for a simulation run where only one vehicle is generated using the distributions and the debug is set to True. The output below is confirmed to be correct.

```
C:\Users\bglow\Desktop\CX-4230-Project-2\des>python des_simulation.py
2.0: Stoplight at Intersections.ELEVENTH has changed to RED. There were 0 waiting.
3.0: Stoplight at Intersections.FOURTEENTH has changed to RED. There were 0 waiting.
4.0: Stoplight at Intersections.TENTH has changed to RED. There were 0 waiting.
4.0: Stoplight at Intersections.TWELFTH has changed to RED. There were 0 waiting.
5: Vehicle(enter_time=5 velocity=17 entrance=1 exit=6 direction=False lane=False) has entered the simulation.
5: Vehicle 0 has arrived at Intersections.TENTH. The vehicle will pass through.
9.002: Vehicle 0 has departed Intersections.TENTH.
26.24: Vehicle 0 has arrived at Intersections.ELEVENTH. The vehicle is waiting.
41.3: Stoplight at Intersections.TWELFTH has changed to GREEN. There were 0 waiting.
50.7: Stoplight at Intersections.FOURTEENTH has changed to GREEN. There were 0 waiting.
55.1: Stoplight at Intersections.TENTH has changed to GREEN. There were 0 waiting.
59.0: Stoplight at Intersections.ELEVENTH has changed to GREEN. There were 1 waiting.
66.868: Vehicle 0 has departed Intersections.ELEVENTH.
83.407: Vehicle 0 has arrived at Intersections.TWELFTH. The vehicle will pass through.
86.372: Vehicle 0 has departed Intersections.TWELFTH.
86.9: Stoplight at Intersections.FOURTEENTH has changed to RED. There were 0 waiting.
91.6: Stoplight at Intersections.TENTH has changed to RED. There were 0 waiting.
100.518: Vehicle 0 has arrived at Intersections.THIRTEENTH. The vehicle will pass through.
102.1: Stoplight at Intersections.ELEVENTH has changed to RED. There were 0 waiting.
103.186: Vehicle 0 has departed Intersections.THIRTEENTH.
103.8: Stoplight at Intersections.TWELFTH has changed to RED. There were 0 waiting.
116.993: Vehicle 0 has arrived at Intersections.FOURTEENTH. The vehicle is waiting.
134.6: Stoplight at Intersections.FOURTEENTH has changed to GREEN. There were 1 waiting.
141.1: Stoplight at Intersections.TWELFTH has changed to GREEN. There were 0 waiting.
142.209: Vehicle 0 has departed Intersections.FOURTEENTH (Exit Point).
142.209: Vehicle 0 has left the simulation.
Statistics:
Total Runtime: 0.005049943923950195 seconds
Total Time for 1 vehicles to exit the system: 142.20884328084387 seconds
Total Duration (Simulation Time): 203.6 seconds
Number of Events: 26
Vehicles Entered: 1
Vehicles Departed: 1
Minimum Travel Time: 137.20884328084387
Maximum Travel Time: 137.20884328084387
STD of TT: 0.0
Average Travel Time: 137.20884328084387
```

5.2.2 Cellular Automata

To verify this model, firstly we ensured the model was programmed in an object oriented way, giving individual classes and functions a single responsibility, to improve readability and comprehensibility of the program. Beyond this, we created an animation of the model, where we print the state of the model to the console window in a way that visually displays the traversal of vehicles through the road corridor. By looking at this and testing many times using different amounts of vehicles, especially with different speeds and different densities per lane, we saw that the model rules for changing speed, advancing, and changing lanes all worked as expected. After running some experiments using the NGSIM data, we could clearly see in the data that the time taken for a vehicle to traverse the corridor was very small if the corridor section traversed was small (say from intersection 1 to 2), and was much longer if the section traversed was long (say from 0 to 6).

5.2 Model Validation

5.2.1 Accounting for the Startup Transient

Before doing validation, it is important that we remove the start-up transient from our data such that the output data from our simulation models are not skewed by initial start-up values. To do this, we used Welch's method in order to better visualize the startup transient. This function is located in the script *welch_avg.py* located within the repo, and accepts an array of vehicle departure times as well as vehicle travel times.

For the cellular automata model, when running a simulation with 10,000 time-steps, it was clear from Figure 5.1, showing the simulation time versus the traversal time, that the traversal times of the vehicles in the simulation were very choppy. In order to see the start-up transient, we used Welch's method with a window size of 100 samples to effectively smooth out the data. This windowed average versus simulation time is shown in Figure 5.2, where we see that when the simulation time reaches approximately 500, the data begins to smooth out and reach more or less a "steady-state". Therefore, for the validity of the model, any statistics gathered before simulation time 500 were thrown out.

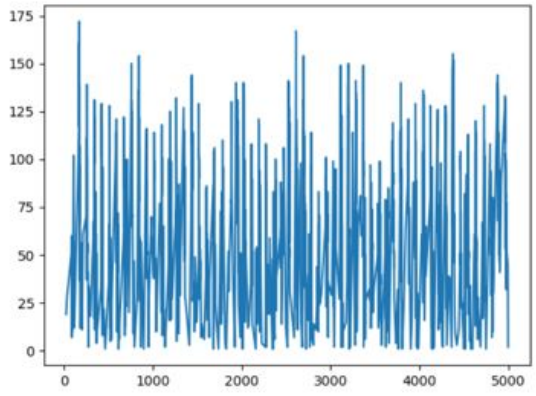


Figure 5.1 Choppy Data

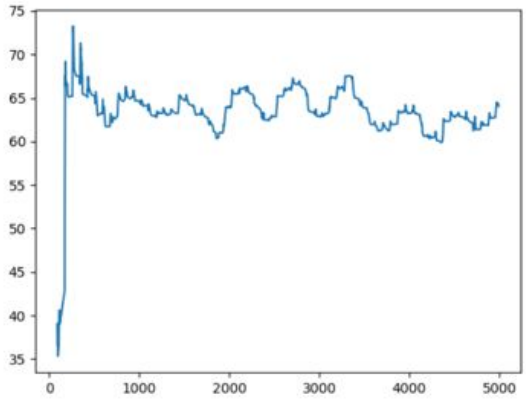


Figure 5.2 Welch's Average

A similar computation using Welch's Method was made for the discrete event simulation. Using our simulation parameters, we set the *max_departures* to 1000 vehicles and let the simulation run. Based on the example results depicted below as well as five other runs, the model typically begins to smooth out after simulation time 700 (which represents a little over 10 minutes). Therefore, we will only consider results and travel times that are collected after this period to account for this start up behavior.

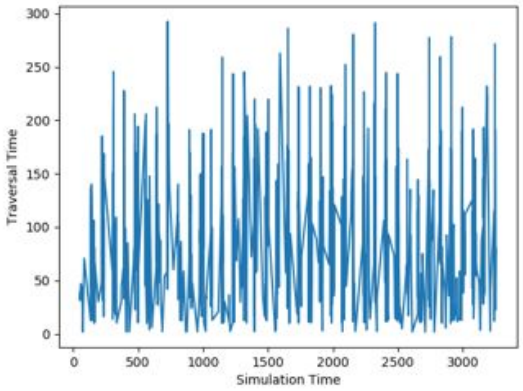


Figure 5.3: Choppy Data for DES

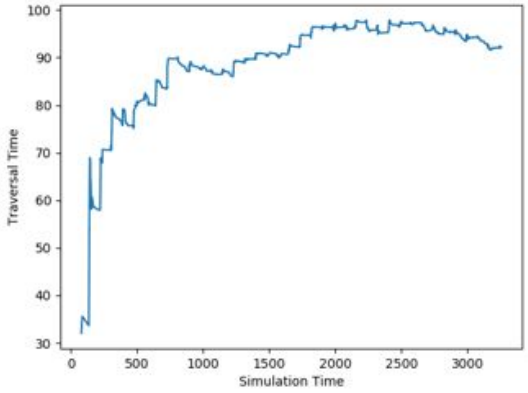


Figure 5.4: Welch's Average for DES

5.2.2 Comparing to the Validation Set

In order to validate these models, we first check our simulations' abilities to produce output statistics that match up with reality, namely the data produced in the NGSIM study, which monitored the actual section of road we are attempting to simulate.

As mentioned previously, we split the NGSIM data into a validation set and a model set, ensuring our model is being compared against test data that wasn't used in the creation of the model. Then, to more accurately compare the NGSIM data against our model, we first removed some outliers from the NGSIM data. These outliers include: vehicles that arrive into the model at intersection 5 (we removed these from the potential inputs to our models, so it makes sense to remove these here), vehicles whose average velocity is zero, vehicles whose entrance intersection is equal to the exit intersection (these would never traverse our models. These were also cleaned out of the input parameter sets as discussed in Section 3.1), and vehicles whose traversal time is one second or less (these vehicles must have not been captured well by the NGSIM study as one second is an unrealistic time to traverse any section of the corridor).

Using this cleaned version of the NGSIM data, the average traversal time from the validation set was calculated to be 91.2 seconds, with a standard deviation of 61.48 seconds. These values are a good representation and standard of comparison for our output statistics. Therefore, we compare our models and how well their average and standard deviation of traversal times match up to the NGSIM data results.

To compare this with the results of our simulation models, first we must ensure that the model parameters match that of the NGSIM data. Most of this was handled in the input analysis section, making sure the vehicle interarrival times match the empirical distribution discovered in the model set of the NGSim data. However, we also ensure that our other model parameters match the NGSIM data. Specifically, the level of traffic was not scaled to a shorter or faster interarrival time as we will do later in other experiments, the road lengths used match those of the real street corridor, and the signal timings match that of the real intersections.

Upon running the simulations with these parameters, and averaging the results over five iterations of each simulator, we took the average vehicle traversal time and standard deviation of the vehicle traversal time to compare with the NGSim values. The cellular automata gave an average vehicle traversal time of 90.07 and a standard deviation of 45.48. The discrete event simulation gave an average vehicle traversal time of 94.97 seconds and a standard deviation of 58.45 seconds.

Here we see that the values of each simulator is reasonably close to the NGSim test data values for mean and standard deviation of the vehicle traversal time, so we can be reasonably sure our models are valid representations of the real system when it comes to the output parameter in question, vehicle transit times.

6 - Experiments and Output Analysis

Now that we have verified and validated our simulation models, we can proceed to perform experiments with the data we collect, and analyze the results. To do this, we will vary the input parameter of traffic density (low, medium, and high), and display the confidence intervals for the mean and standard deviation of vehicle traversal times. That is to say that we will, from each run of a simulation, collect the average and standard deviation of the vehicle traversal times for that run, then for several iterations of these runs, we will produce confidence intervals on the normally distributed random variables mean transit time and standard deviation of transit times.

First, we compute the mean and standard deviation of vehicle transit times for each simulation over ten runs. Then using these means and standard deviations, for both normally distributed random variables mean and standard deviation, we calculate the sample mean and sample standard deviation. Then using these values, we calculate a 95% confidence interval for each using the student's t distribution with four degrees of freedom.

6.1 Confidence Intervals for Different Traffic Levels

For the cellular automata with normal traffic, the confidence interval for the mean is (74.01, 91.97), where the mean of the mean vehicle transit time is 82.99. The confidence interval for the standard deviation is (38.58, 50.66), where the mean of the standard deviation vehicle transit time is 44.62. This information is graphically displayed in figure 6.1, where the bar indicates the mean of the random variable, and the black line indicates the confidence interval for that random variable.

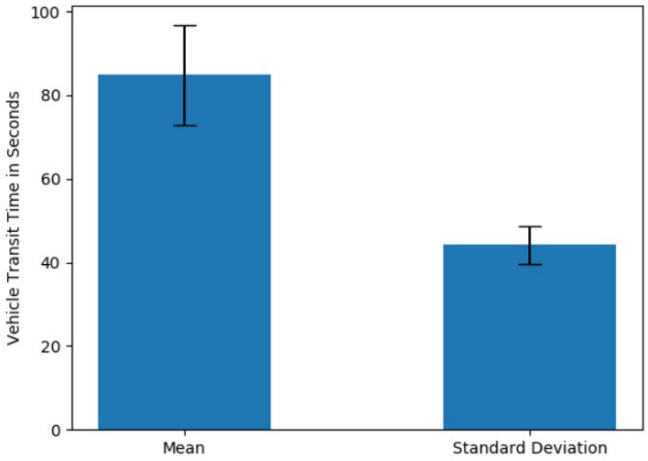


Figure 6.1 Confidence Intervals on Mean and Standard Deviation with Medium Traffic

For the discrete event simulation with normal traffic, the mean vehicle transit time was 94.45 with a 95% confidence interval of (86.60, 102.30), and mean standard deviation of 58.78 with a 95% confidence interval of (54.67, 62.89). This information is graphically represented in figure 6.2.

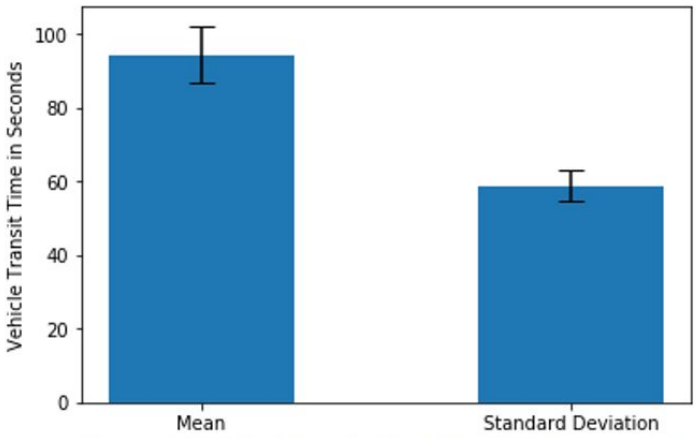


Figure 6.2 Confidence Intervals of Mean and Standard Deviation on the DES model with Medium Traffic

Remembering from earlier in the model validation section, the mean vehicle transit time based on the NGSim test data was 91.2 seconds, with a standard deviation of 61.48 seconds. Here we see that the cellular automata model gave a 10 second shorter value and the discrete event simulator was only 3 seconds longer at 94 seconds. From this first test, the discrete event simulator seems to match the real world data more, and give a longer mean time and a higher standard deviation. Also notable is that the confidence intervals on the discrete event simulation are smaller, indicating that it produces more consistent results.

For the case of light traffic, we used a scale factor on the vehicle interarrival time of 1.3, making the number of cars entering the model much lower over time. For this scenario using the cellular

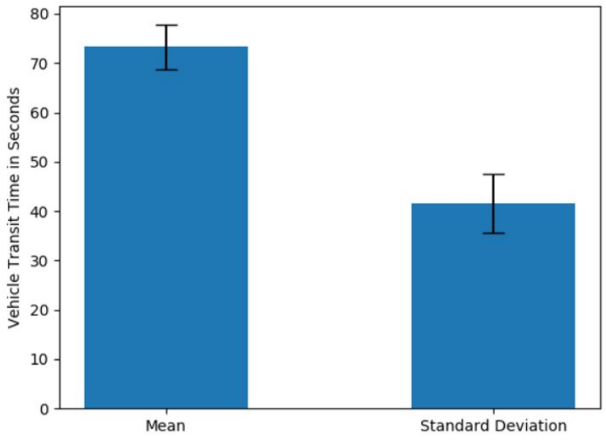


Figure 6.3 Confidence Intervals on Mean and Standard Deviation with Low Traffic

automata simulation, the mean value of the mean vehicle transit time was 73.28 with a 95% confidence interval of (68.83, 77.74), and the mean of the standard deviation was 41.49 with a 95% confidence interval of (35.54, 47.45). This is graphically shown in figure 6.3.

For discrete event simulation in the scenario of lower traffic, the mean vehicle transit time was 88.39 seconds with a 95% confidence interval of (83.85, 92.94), and a mean standard deviation of 56.16 with a 95% confidence interval of (52.97, 59.35).

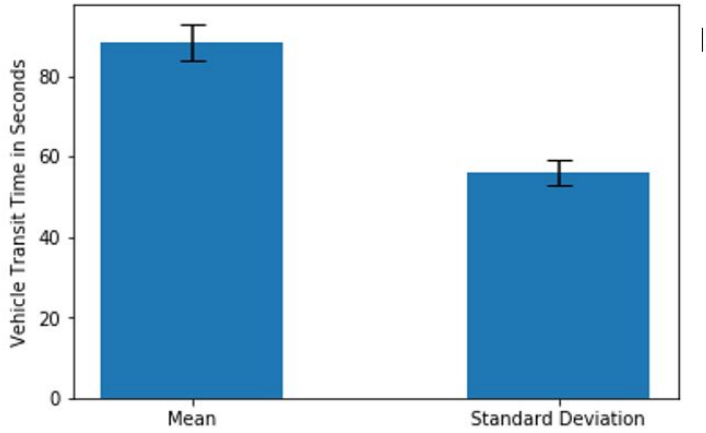


Figure 6.4 Confidence Intervals of Mean and Standard Deviation on the DES model with Light Traffic

Based on figures 6.3 and 6.4, we see that the reduction in traffic affected the models in similar ways. In each model, the mean vehicle traversal time decreased by around 10 seconds, and the confidence intervals for both statistics decreased significantly as well. This indicates that due to a lower traffic load, vehicles were able to traverse the model roads faster than in the medium traffic load, as one would expect. Also, the reduced confidence interval on the mean vehicle transit time indicates that both models experienced more consistent results, showing that higher traffic may be correlated with a more randomly varying vehicle traversal time.

For the case of high traffic, we used a scale factor on the vehicle interarrival time of 0.7, increasing the number of vehicles entering the simulation significantly. For this scenario using the cellular automata simulation, the mean value of the mean vehicle transit time was 250.03 with a 95% confidence interval of (178.23, 321.83), and the mean of the standard deviation was 115.66 with a 95% confidence interval of (88.06, 143.26). This is shown graphically in figure 6.5.

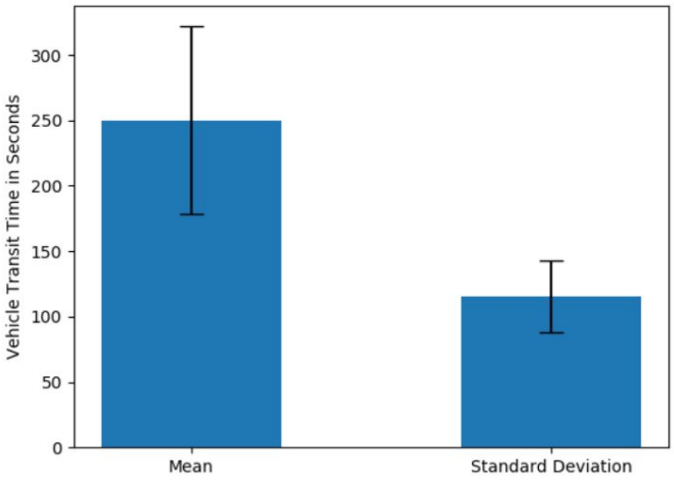


Figure 6.5 Confidence Intervals on Mean and Standard Deviation with High Traffic

For the discrete event simulation, using the high traffic scale factor of 0.7, the mean value of the mean vehicle transit time was 161.61 with a 95% confidence interval of (138.07, 185.15), and a mean standard deviation of 125.67 with a confidence interval of (98.08, 153.26). This information is shown graphically in figure 6.6.

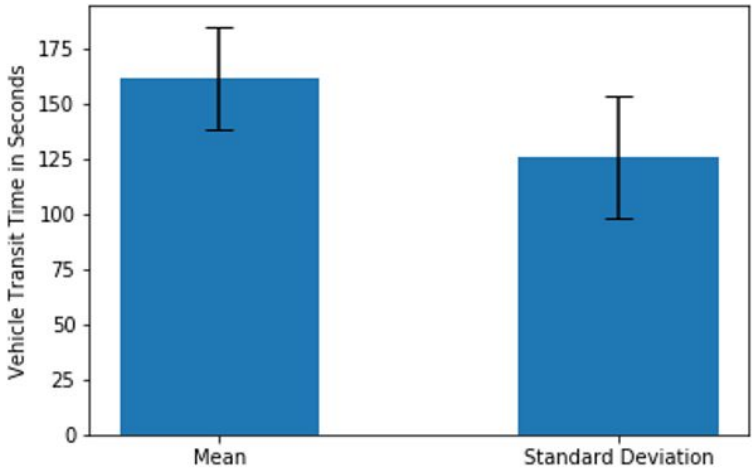


Figure 6.4 Confidence Intervals of Mean and Standard Deviation on the DES model with Heavy Traffic

In this last scenario of high traffic, we see that both models show a drastic increase in mean and standard deviation of the vehicle transit times. This makes sense since we would expect more vehicle arrivals to increase traffic. Something else we also see however is that the confidence intervals increased significantly as well. This indicates that the results of the individual tests were much more erratic than for the medium or light traffic. Something we postulated after the light traffic example was that a change in confidence interval size is positively correlated with

traffic, here this is shown again. One explanation for this is that due to higher levels of traffic, the behavior of the vehicles is more volatile and subject to significant delays. For example, if a car randomly slows down one unit of speed in the cellular automata model, it stands to affect many more vehicles than when there is low traffic.