Final Report

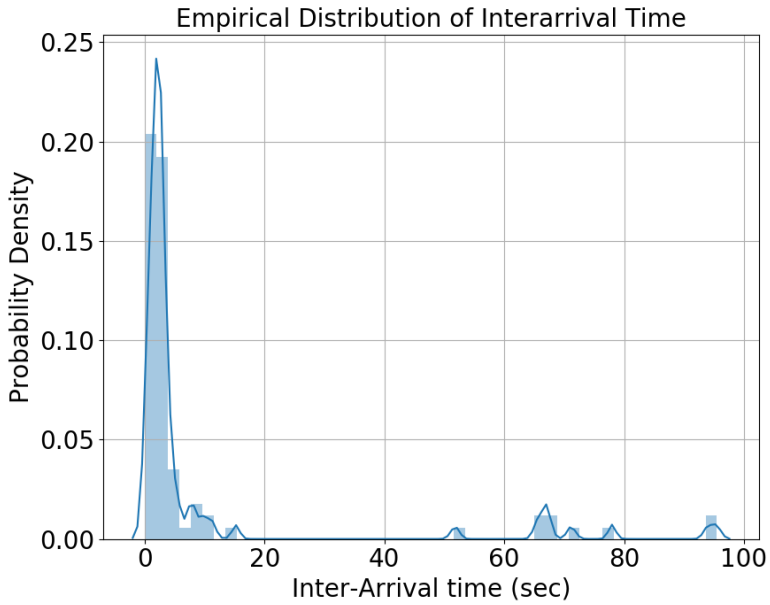
Group 59
Member: Tianyang Cao, Yanmo Xu, Yuhao Chen
GitHub Repo: https://github.gatech.edu/tzheng34/CSE6730-Group-59-Project2

Problem Statement

Our simulator development aims to calculate the average travelling time for vehicles to traverse a portion of Peachtree Street, the corridor from 10th to 14th street. We will simulate the traffic in one direction. Three conceptual models are implemented: event-oriented queueing model, activity-scanning queueing model, and the cellular automata model. The goal of the simulation is to generate the distribution of vehicle travel times as they traverse this stretch of Peachtree Street for different levels of traffic. Different parameter settings will be analyzed in the study as well.

Input Analysis

We use the NGSIM data to fit an empirical distribution for interarrival time for traffic entering the road network. Because we only care about the traffic from south to north, so we only use the cars entering the region from the first intersection. So we use half of the data in our dataset to create our empirical distribution and use the remaining data to validate our simulation models.



Once we have the empirical distribution of the entrance, we can use rejection-acceptance method to generate random interarrival time.

We generated traffic with different incoming flow rates in the following way. The empirical distribution above was used as the regular/intermediate traffic level. To increase the flow rate, we first generated random numbers from the same original distribution. Then for each

generated inter-arrival time value, we subtracted some amount from it (set it to 0 if it became negative) and used the new value as the final inter-arrival time input. The subtraction amount is proportional to the increase amount of the flow rate. The visual effect of this approach is shifting the entire empirical distribution to the left, hence reducing the overall inter-arrival time values. To decrease the flow rate, we did the opposite procedure. For validation, we recorded the average number of vehicles injected into the road during a fixed time interval (computed with multiple simulation runs) while shifting the empirical distribution from left to right. The number of injected vehicles kept decreasing as the distribution was shifted further to the right. This showed that our approach to change the incoming traffic flow rates is valid.

Simulation Model #1. Cellular Automata

I. Conceptual Model Description

The road corridor under study is divided into small individual cells. Each lane is represented by an array of cells. Each cell has only two states: empty or occupied by a vehicle. For each vehicle, it has several properties such as position, speed, driver aggressiveness, etc. The system is updated at each discrete time step, based on the designed update rules. These rules characterize the different factors on the road (traffic lights, turning at intersections, lane changing, vehicle speed changes, etc.) that affect the overall traffic condition. Each vehicle's states are updated by those carefully designed rules. This CA model is built to simulate the traffic and estimate the average travelling time on the target corridor.

Model Assumptions:

1. All vehicles have the same length and acceleration rate
   Justification: from the dataset the majority of vehicles have similar length, so taking their average length as the length of all vehicles is reasonable. Also, most cars change their speeds at similar rates (car performance specs have little effects on the target road segment).
2. A vehicle can only move among different cells, meaning that its position change is discrete instead of continuous (each cell represent a fixed size segment of the road)
   Justification: on the macro scale, vehicles can be viewed as moving in a cell grid. The discretized position change is a reasonable simplification of the actual continuous change.
3. Since position changes are discrete, the vehicle speed level is also discrete (each level allows the vehicle to move exactly to different cell locations)
4. All vehicles' accelerations are smooth (no abrupt increase in speed), while abrupt braking is allowed (sharp decrease in speed is permissible)
   Justification: this pattern can be observed from the sample dataset, and it makes sense for the relatively short road corridor under investigation.
5. New vehicles can only enter the target road corridor from the first intersection, but can exit at any intersections
   Justification: again, this is the general pattern observed from the dataset.

Model Limitations:

1. Due to assumption 1 above, if in some cases where there're non-negligible variations in vehicle lengths, then the model simulation results won't be very accurate.
2. Due to assumption 5 above, if there are a non-negligible number of vehicles entering into the target corridor at the intermediate intersections, then the simulation result would be inaccurate (would probably produce a shorter average travelling time than the actual condition)
3. If there are emergency situations on the road, such as accidents, traffic regulation, that could affect the vehicle normal movement patterns, then naturally the simulation results would not be accurate

Otherwise, this simulation model is capable of producing reasonably accurate results.

Model Parameters and Setup:

There are a list of model parameters to be determined from sample traffic data. First, the dataset shows that the average vehicle length is 16 feet. Therefore, in the model each cell corresponds to 16 feet in real world. Then, The actual length of each road section (obtained from data) is used to calculate the number of cells that represent each section. The traffic lights are set at the corresponding cell location as well, with the timing information obtained from the dataset. The maximum speed limit for the target corridor is 51 ft/s, which is converted to 4 cell length per second. New vehicles arrive at the first intersection based on the empirical distribution rate as discussed in the Input Analysis section.

Model Implementation:

The simulation model is written in Python. The key part of this model is the object class that represents a vehicle (called 'Vehicle' class below). This object class has several attributes such as the vehicle's current cell position, current speed, new speed at the next time step, preceding and trailing vehicle pointer, etc. The two class functions are able to calculate the vehicle's new speed at the next time step and to update the vehicle's position accordingly. Each vehicle on the road is an instantiation of the 'Vehicle' class and is linked with each other to form a linked list data structure.

The 'Vehicle' class function to calculate the new speed plays a huge role in the traffic dynamics, so this paragraph will be dedicated to its design. For each vehicle, the following set of rules are carried out sequentially to determine its new speed at the next time step (based on its current speed):

1. The vehicle increases its speed by 1 (unit: number of cells per second) if it hasn't reached the maximum speed limit yet
2. The vehicle maintains its speed with 40% probability (if so, undo the first step)
3. The vehicle decreases its speed by 1 with 30% probability, or stays at rest if the current speed is already 0 (if so, undo the first two steps)

4. If the vehicle's new speed (obtained from the above steps) is larger than the distance from the preceding vehicle, set the new speed to be the inter-vehicle distance
5. If the vehicle is currently at a road intersection, set the new speed to 0 if the traffic light is red, or maintain the new speed otherwise
6. If the vehicle is currently at a road intersection, exit the road with 10% probability (by removing it from the vehicles' linked list)

Rule 1 characterizes the driver's desire to always try to achieve the max speed. Rule 2 represents the uniform speed movement. Rule 3 tries to model the irregular braking due to factors like bumpy road condition, erratic driving, etc. Rule 4 corresponds to collision avoidance. Rule 5 is a straightforward response to traffic lights. Rule 6 models the through/turning movement of the vehicle at each intersection. Since the majority of cars on the corridor stay there the entire time based on the sample dataset, the probability of turning is set to only 10% to reflect the pattern. Finally, the update position function for the 'Vehicle' class simply adds the vehicle's new speed with its current position to get the new position.

The flow of the program is the following. There is a main loop that keeps iterating (advancing time steps) until some user defined termination condition. Initially, the entire corridor is empty. At each time step, if the first intersection (new vehicle entrance) is empty, a new 'Vehicle' object is instantiated (or not) based on our pre-fitted Poisson distribution of vehicle inter-arrival time. This new vehicle is linked to its preceding vehicle and becomes a new node in the existing vehicle linked list. Then, the main loop calls each existing vehicle's class functions to calculate their new speed and update their cell positions accordingly. Especially, in the new speed calculation function, if a 'Vehicle' object decides to turn at an intersection, its neighbors in the linked list will be connected properly. Finally, the main loop checks if any vehicles reach/pass the very end of the corridor. If so, they will be popped out of the linked list and their total travelling times on the corridor are used to update the average travelling time.

II. Verification and Validation

1. Verification

   To verify the software code functions correctly as the simulation model specifies, the following steps are carried out:

   a. During both the transient period and the steady state of a simulation run (several runs are performed to ensure the validity of the results), numerous batches of vehicles were randomly chosen to be investigated. Their entire trajectories (including their position, speed, movement pattern at each time step) were studied and verified to match the desired behavior.
   b. Over the entirety of a simulation run (again several runs are performed), vehicle states at each intersection were recorded. The results demonstrated the correct functioning of the traffic lights as well as the through/turning movement patterns (the ratio of the number of cars going through to those turning matched well with the probabilities we set in the model).

c. The red light times at all intersections were increased gradually, and the simulated average travelling time (steady state value) increased proportionally as well. Similar results were obtained for changing the green light times. This served as another way to verify the simulation model.
d. The maximum speed for the corridor was raised up and down, and the average travelling time increased or decreased in the expected way. This verified the correct implementation of the vehicle driving behavior.
e. Increasing/decreasing the probability of maintaining (or reducing) the current speed as opposed to speeding up for any vehicle led to an increase/decrease in the simulated average travelling time. This verified the correct implementation of the vehicle driving behavior.

2. Validation

The following steps were taken to compare the simulation model with the actual traffic conditions. As mentioned in the Input Analysis section, half of the sample data was used to fit the empirical distribution. The other half of the dataset was used for validation here. We used it to fit and generate another empirical distribution of the vehicle inter-arrival times. Then this distribution was used as the input to the simulator. Multiple simulation runs were performed to obtain the average travelling time result. Finally, we calculated the actual average travelling time of those vehicles in the target half dataset and compared it with the simulation result. They were reasonably close so the validation step is complete.

III. Experiment Design and Output Analysis

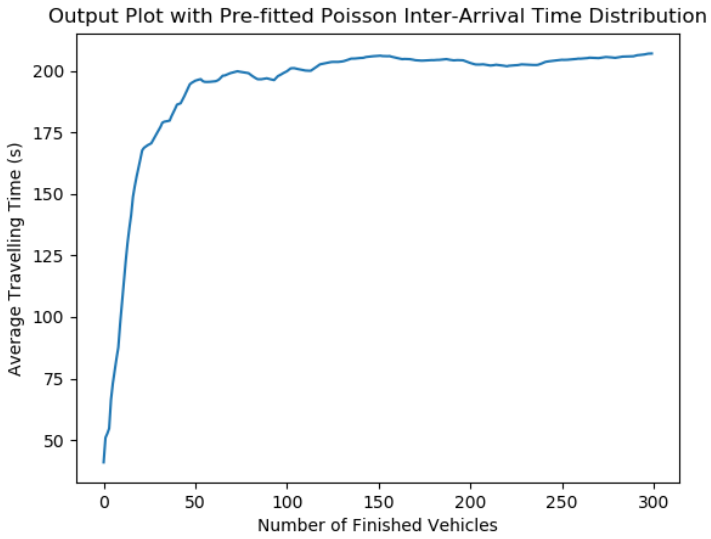
The output of a simulation run is the average travelling time. In order to identify the transient (warmup) period of a simulation run, a plot is created to show the current average travelling times right after each additional vehicle finishes travelling the entire corridor. The x axis represents the number of vehicles that have already finished the corridor, and the y axis shows the computed average travelling time computed from that amount of finished vehicles.

Number of Simulation Runs For Each Data Point:

Different simulation runs have slightly different results due to the random number generators in the code (inter-arrival time of new vehicles, turning at each intersection, individual vehicle speed changes). Therefore, 5 simulation runs (with the same distributions for all random number streams) were performed and the output results were averaged to obtain the final output plot. This number of runs was determined in the following way. We started with one run and kept adding more runs while obtaining an average output plot after each additional run. When the average output plot stopped having noticeable changes, the process stopped and the number of simulation runs performed so far became the number of runs we used for each data point.

Identification of the Initial Transient (Warmup) Period:

Below is the output plot when our pre-fitted Poisson distribution was used to generate the inter-arrival time for new vehicles:



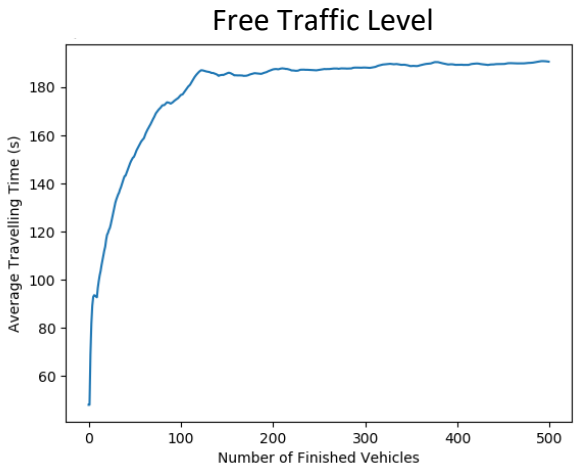
From the plot, we can see that the output curve doesn't have much oscillations/changes once it reaches its steady state. Therefore, it is feasible to distinguish the initial transient period and the steady state period by carefully looking at the plot (without using Welch's Moving Average Method). For the plot above, the steady state is reached when around 150 vehicles have finished travelling the entire corridor.

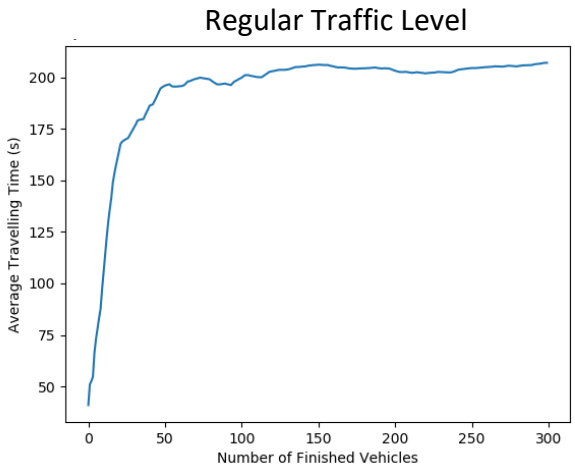
Confidence Interval:

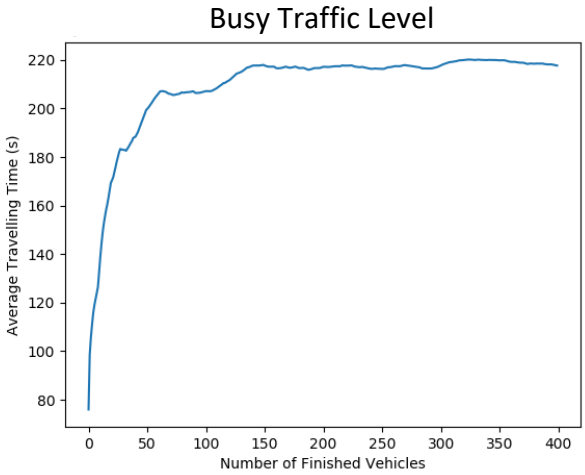
5 times: mean=291.3 , confidence interval = [290.1, 292.5]

Output Results for Different Traffic Level:

The output plots for three different traffic levels (busy, regular, free) are shown on the next page. The incoming traffic flow rates were changed using the techniques discussed in the Input Analysis section.







For all three traffic levels, the simulation steady state is reached when around 150 vehicles have finished travelling the corridor. When the traffic level is free, the steady-state average travelling time is 180 seconds (or 3 minutes). When the traffic level is regular, the average travelling time is 200 seconds (or 3.3 minutes). When the traffic level is busy, the average travelling time is 220 seconds (or 3.7 minutes).

Simulation Model #2. Activity-scanning Queueing Model

I. Conceptual Model Description

Model Assumption:

1. Cars wait in queue when enter the intersection and pass the intersection in the timestamps order.
2. Overtaking can only happen when one car is faster than another. No car accident. No

emergency vehicles.
3. Cars can only pass the intersection when the light is green

Model Simplification:

1. Cars can only enter the region from the southern side of Peachtree Street.
2. Cars can only exit the region from five intersections.
3. Each car is a point (no length, no width).
4. Each car has an average speed and doesn't change speed when passing the whole region.
5. All the traffic lights have the same signal timing.
6. Each car already knows which intersection it will exit before entering the region.
7. No turn right or left. If a car wants to exit in an intersection, it will directly be removed from the intersection

Model Limitation:

1. One car may pass different roads and intersections with different average speed.
2. Cars can enter the region from the intersections.
3. The width and length of cars may affect the traffic

Model Implementation:

My simulation has four different activities: EnterRegion, OnTheRoad, EnterCross, OnTheCross. Each activity contains the entity(car) information, start time and end time of the activity. There are 6 sections of roads and 5 intersections in my model. Each road and intersection have their own priority queue. Simulation time is discretized into time steps. A time step is one second. For each time step, scan the lists of activities.Above are the simulation steps.

Step 1: Create car entity:

When a car enters the region, create a car entity which includes car_number, arrival_time and car_speed.

Step 2: Create activity_EnterRegion

When a car entity is created, create a activity_EnterRegion(Car). Set the end_time=car.arrival_time.

Step 3: Create activity_OnTheRoad

When <activity_EnterRoad> is created or <activity_OnTheCross> is end, create a activity_OnTheRoad(activity_EnterRoad,Road).

Set start_time= activity_EnterRoad.end_time and end_time=start_time+road.length/car.speed. Send it into activity list.

Step 4: Create activity_EnterCross

When simulation time = activity_OnTheRoad.end_time, create a <activity_EnterCross>. Send it into a queue.

Step 5: Check the traffic lights

If the traffic light is green or yellow, pop the first element in the queue.

Step 6: Create activity_OnTheCross

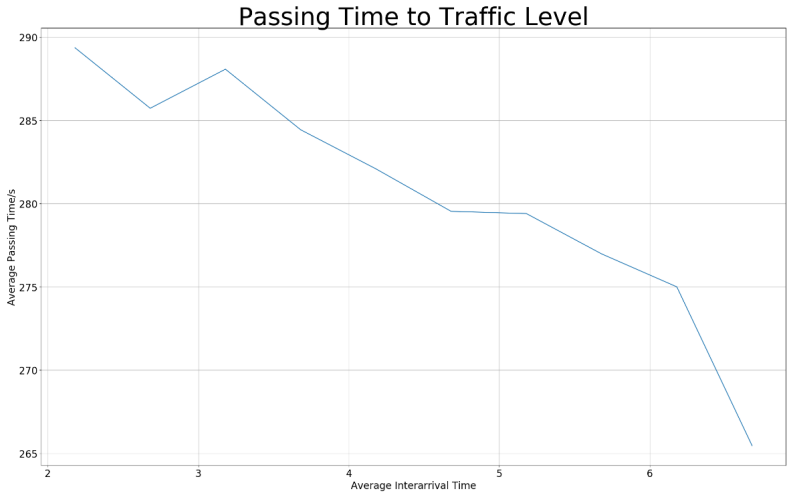
If there is an element popped from the queue, create a <activity_OnTheCross(activity_EnterCross,Cross)>.Set start_time= simulation

time+3*lenth(queue), end_time=start_time+cross.length/car.speed. Send it into activity list.The traffic in the queue will course more time to pass the intersection.

II. Verification and Validation

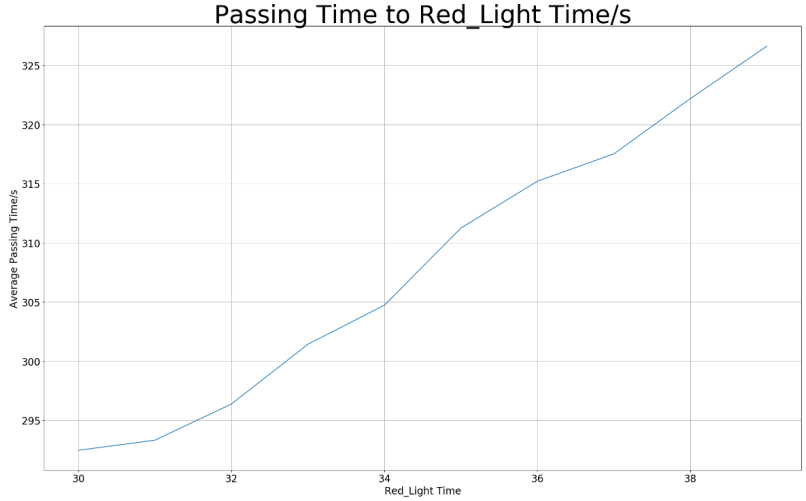
1. Verification:

a) When the interarrival time at the entrance increases, the average passing time should increase. Because the traffic level increases.



When the average interarrival time increases, the average passing time increases.

b) The red light duration at all intersections were increased gradually, and the simulated average travelling time (steady state value) increased proportionally as well. Similar results were obtained for changing the green light times. This a way to verify the simulation model.



When the duration of red light increase, the average passing time increases.

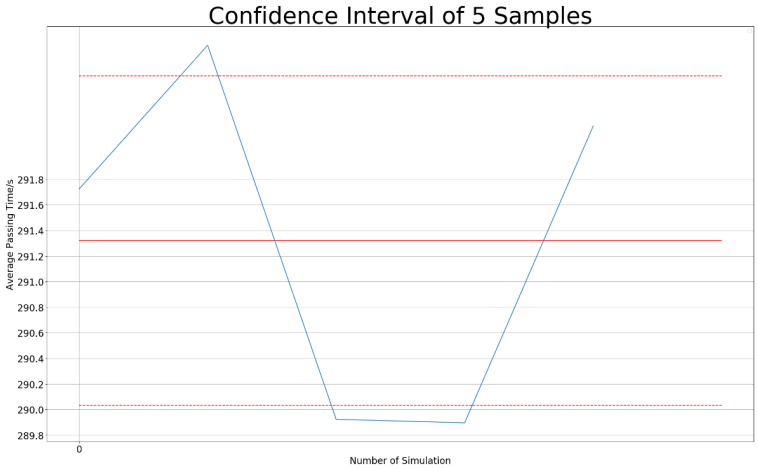
2. Validation:

The following steps were taken to compare the simulation model with the actual traffic conditions. As mentioned in the Input Analysis section, half of the sample data was used to fit the empirical distribution. The other half of the dataset was used for validation here. We used it to fit and generate another empirical distribution of the vehicle inter-arrival times. Then this distribution was used as the input to the simulator. Multiple simulation runs were performed to obtain the average travelling time result. Finally, we calculated the actual average travelling time of those vehicles in the target half dataset and compared it with the simulation result. They were reasonably close so the validation step is complete.

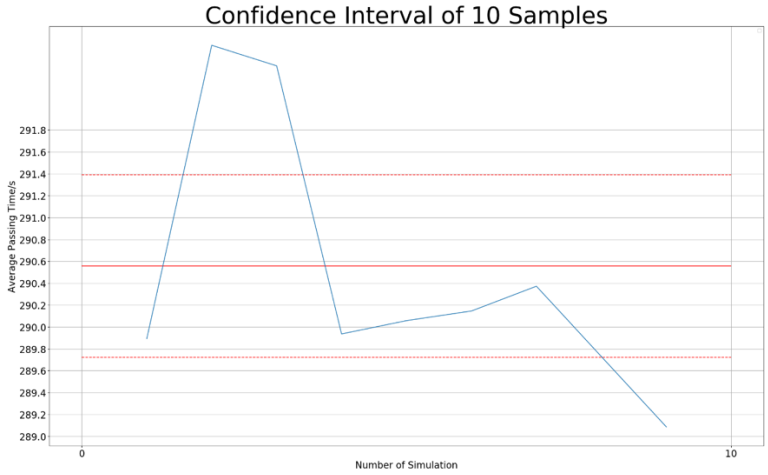
III. Experiment Design and Output Analysis

Confidence Interval:

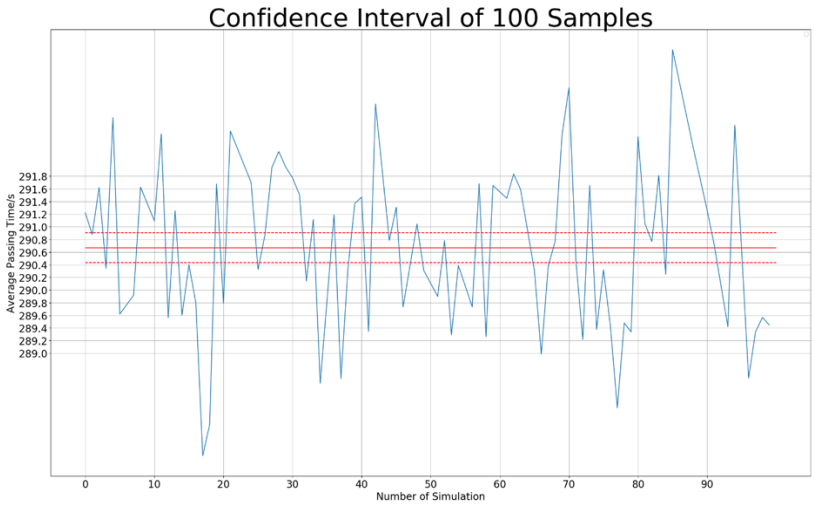
5 times: mean=291.3 , confidence interval = [290.1, 292.5]:



10 times: mean=295.5, confidence interval=[289.7, 291.4]:



100 times: mean=290.65, confidence interval=[290.4,291]:

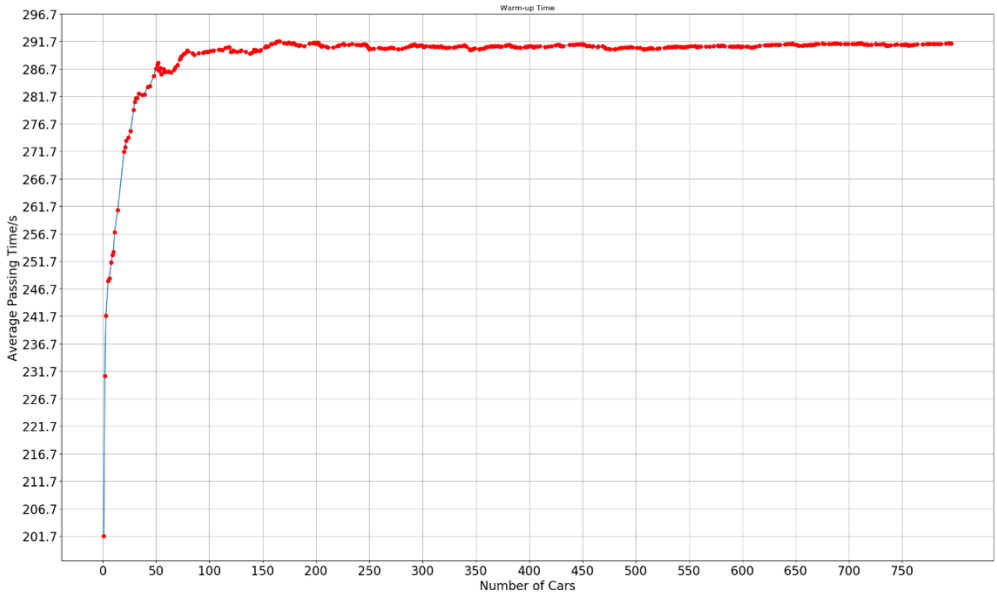


Number of Simulation Runs:

10 times. When we run the simulation 10 times =, the confidence interval is very close to the confidence interval of 100 times simulation.

Warm-up Period:

As we can see from the figures, after 150 cars, the average passing time tend to be stable around 291.7.



Simulation Model #3. Event-oriented Queueing Model

I. Conceptual Model Description

The process of passing the 10th to 14th traffic system is divided into three types of event, which defined in the python file: arrive, wait for the green light and running on the road, every event would links to another event of one certain car or invoke another car to start simulation, for the Car class, it has several attributes help us to proceed the process and by updating these parameters.

```
def __init__(self, onetime, carindex, alight, onelist, onestr, onenumber, number,vcount):
    self.Curr_time = onetime
    self.Index = carindex
    self.Curr_traffic_light = alight
    self.arrivetime = onelist
    self.Event = onestr
    self.Processtime = onenumber
    self.cap = number
    self.valid = 1
```

The system is updated at each trans step between events, based on the designed update rules. These rules characterize the different factors of the event the car is processing (traffic lights, turning at intersections, the time slot) that affect the overall traffic condition. Each vehicle's states are updated by those carefully designed rules. This event-oriented model is built to simulate the traffic and estimate the average travelling time on the target corridor.

Model Assumptions:

1. All vehicles has the same length and same time to pass all the sections as defined
   Justification: from the dataset the majority of vehicles have similar length, so taking their average length as the length of all vehicles is reasonable. Also, most cars change their speeds at similar rates (car performance specs have little effects on the target road segment).
2. A vehicle can only in one direction, that is from 10th to 14th , no car travels in a reversed direction (each cell represent a fixed size segment of the road)
   Justification: as asked we did not employ any car travel in a reversed direction in this simulation problem
3. The car could be out to like 13th street in some probability, here we set it like 0.1, scan whether the car would be out when every event called waiting for light.
   Justification: observed from datasets, very few samples would leave the system in the middle process
4. All vehicles' accelerations are smooth (no abrupt increase in speed), while abrupt braking is allowed (sharp decrease in speed is permissible)
   Justification: this pattern can be observed from the sample dataset, and it makes sense for the relatively short road corridor under investigation.
5. New vehicles can only enter the target road corridor from the first intersection, but can exit at any intersections

Justification: again, this is the general pattern observed from the dataset.

Model Limitations:

1. Due to assumption 1 above, if in some cases where there're non-negligible variations in vehicle lengths, then the model simulation results won't be very accurate.
2. Due to assumption 5 above, if there are a non-negligible number of vehicles entering into the target corridor at the intermediate intersections, then the simulation result would be inaccurate (would probably produce a shorter average travelling time than the actual condition)
3. If there are emergency situations on the road, such as accidents, traffic regulation, that could affect the vehicle normal movement patterns, then naturally the simulation results would not be accurate
4. Due the logical process in the codes, the traffic level we can simulate is limited, however, we could solve it by running on the pace, no limit of car number being observed

Otherwise, this simulation model is capable of producing reasonably accurate results.

Model Parameters and Setup:

There are a list of model parameters to be determined from sample traffic data. Before the simulation, we set some factors that can effect the consumed time in the process as below:

```
already_pass_cars = [0, 0, 0, 0, 0]
# The cars already pass the nth street
curr_car_in_different_path = [0, 0, 0, 0, 0]
# The car numbers on the (n+10)th road
constant_to_pass = [0, 30, 45, 50, 40]
'''
```

```
Through_light_time = 0.8
red_duration = 25
green_duration = 15
```

Model Implementation:

The key to the establishment of this model is the establishment of this object of the vehicle. Many parameters of the vehicle object are used to guide the transition between processes, just like curr_traffic_light indicates the traffic indicator now facing, the whole process monitoring key parameters are used for conversion. To the next event, if it is detected that the current vehicle has gone out halfway or has reached the end point, then we will update the timeline event, and the next car enters the system, that is, waiting for the first traffic light.

We define several functions that drive the process.
(1) Used to judge whether the current traffic light is allowed to pass
(2) Output waiting time for green light
(3) Exit directly at each intersection (excluding the last intersection) with a random probability of 10%
(4) Define the arrival event: update the event label of the vehicle, update the current time of the vehicle, check whether it is the last intersection, check whether it is the last one, and update the number of vehicles that have passed at the intersection before, if the front is not a green light or There is a vehicle waiting before waiting for the green light and the road to go smoothly to the next event (running_on_road)

(5) Define the driving event on the road: update the time stamp of the vehicle, update the current time of the vehicle, update the index of the traffic light facing the direction, enter the waiting green state, know that the waiting is completed and enter the next event (wait)
(6) Define the traffic light events: check the traffic lights, check whether the road conditions are smooth, until both match, enter the arrival event

II. Verification and Validation

1. Verification

To verify the software code functions correctly as the simulation model specifies, the following steps are carried out:(we used the similar approach to do the verification, details differs from model to model)

a. During both the transient period and the steady state of a simulation run (several runs are performed to ensure the validity of the results), numerous batches of vehicles were randomly chosen to be investigated. Their entire trajectories (based on ***event time stamp***) were studied and verified to match the desired behavior.
b. Over the entirety of a simulation run (again several runs are performed), vehicle's parameter at each intersection were recorded. The results demonstrated the correct functioning of the traffic lights as well as the through/turning movement patterns (the ratio of the number of cars going through to those turning matched well with the probabilities we set in the model).
c. The red light times at all intersections were increased gradually, and the simulated average travelling time (steady state value) increased proportionally as well. Similar results were obtained for changing the green light times. This served as another way to verify the simulation model.
d. Change the probity of 'out' at every section, would do impact on the graph we get, due to that more cars would 'leave' let the system not so busy.

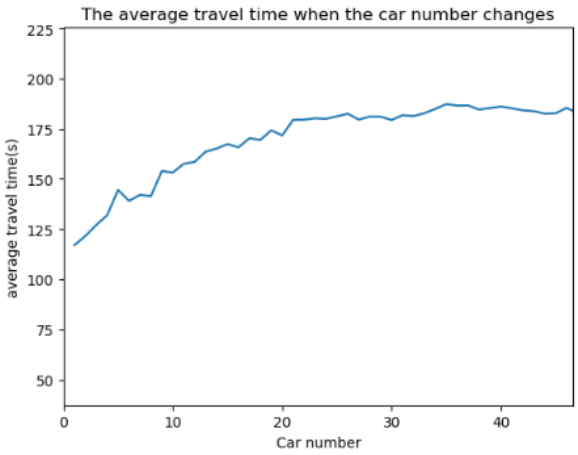
2. Validation

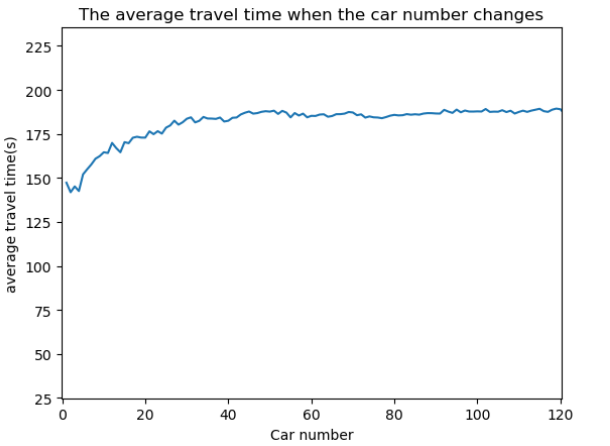
The following steps were taken to compare the simulation model with the actual traffic conditions. As mentioned in the Input Analysis section, half of the sample data was used to fit the empirical distribution. The other half of the dataset was used for validation here. We used it to fit and generate another empirical distribution of the vehicle inter-arrival times. Then this distribution was used as the input to the simulator. Multiple simulation runs were performed to obtain the average travelling time result. Finally, we calculated the actual average travelling time of those vehicles in the target half dataset and compared it with the simulation result. They were reasonably close so the validation step is complete.

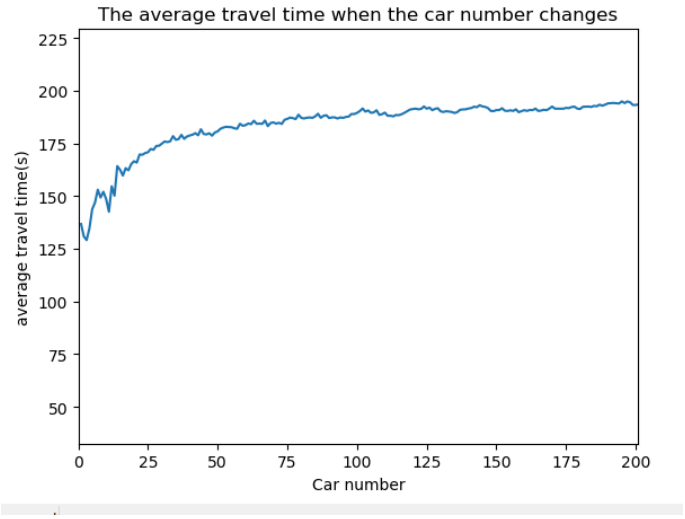
III. Experiment Design and Output Analysis

The output of a simulation run is the average travelling time. In order to identify the transient (warmup) period of a simulation run, a plot is created to show the current average travelling times right after each additional vehicle finishes travelling the entire corridor. The x axis represents the number of vehicles that have already finished the corridor, and the y axis shows the computed average travelling time computed from that amount of finished vehicles.

Output Results for Different Traffic Level:







We set three levels of traffic to ***50/120/200*** vehicles

Identification of the Initial Transient (Warmup) Period:

Here, ***moving-average method*** is used to distinguish between the warm-up period and the stable period, however, without using moving average, we still could find it very easily.

Confidence Interval:

We can see from three graphs that the traveling time get stable after the car numbers comes to 150, and we set ***confidence interval*** as +- 0.5, and we could get alpha as little as 0.05, which is good one.

Number of Simulation Runs For Each Data Point:

I ran ***100 times*** for each data point due to the experiment of testing different numbers of runs, it performed really "randomly" when the number is small, however, 100 is one proper number for repeating the operation.

Model Comparison

Warmup time Comparison:
Cellular Automata:150 cars
Activity scanning: 150 cars
Event oriented: 75 cars

So CA and activity scanning model roughly have the same warm-up period and Event-oriented model have much less warm-up period. Probabally because event-oriented model is the simplest. So it is fastest to reach a stable situation

Simulator Software Code

All three simulator codes are in the GitHub repo. The README file contains all the relevant information.