CS166 LBA

Feedback and Grading

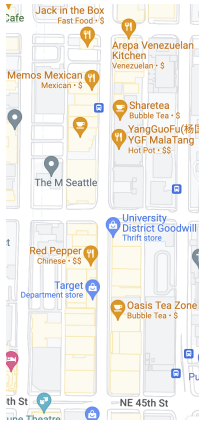
I would like to get feedback for:

1. Code structure: I probably spent way more time than I should have on the code itself. Despite that, I think my code is pretty messy, and because I had so many different components calling one another, the code ends up being pretty tangled and hard to debug. If prof/ TA has time (even 5 minutes) to look at the code and give me actionable feedback, that would be great!
2. Whether my theoretical analysis makes sense, if there are things I should have considered.
3. Potentially why there's a dip for the road density!

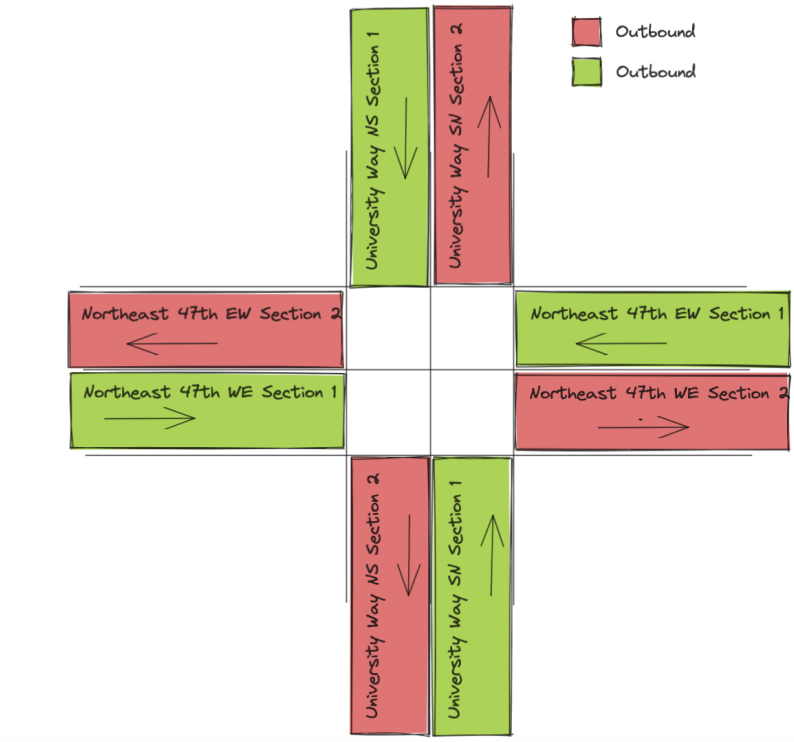
Overview

For break, I went to Seattle for a few days to meet with friends and checked out my place for the summer (but couldn't meet with prof due to time constraint, sorry!). I stayed at University District, and was regularly walking past a busy intersection where university students frequented throughout the time I was there. While the streets are always busy, they are rarely congested. That makes me wonder, what configurations would make the intersection busy? My simulation models after this intersection to find an answer.

Code Assumptions



Original street on Google Maps



Simulated street

To simulate the intersection, I first made several assumptions, some of which are suboptimal, but should suffice for the purpose of the simulation:

1. Most crucially, the actual intersection is not in the simulation. This means that when the car is at the end of the road, and if the traffic light is green, it will **immediately go to the next road** (but will not pass through an intersection inbetween).
2. The road is divided into sections (see the drawing above), so **each section is considered a road on its own**.
3. I also assume that we can **only add cars at the beginning of the road** (in Section 1 of all roads) **at an exact frequency**. This is also not realistic in the real world, because cars can start at any point on the road, and its frequency is more random than deterministic.
4. The traffic light for the intersection only lets cars on 1 road go at any one time.

The rationale for 1, 2, 4 and 5 is to simplify our simulation without compromising on the results, since we are mainly interested in the average traffic density, not the exact coordinates of the car. The main purpose for 3 is to facilitate roads turning into the correct path.

Code Structure

The code is found here.

There are 5 classes in my code:

1. Class `Road` with the name, length, next roads and intersection. It stores all the cars in the road in a `queue` object. From this class, we can add a new car to the road, get the roads that it is connected to, and when updating, it would move all the cars in the road forward.
2. Class `Car` with the road, index on the road, speed, turn directions, and probability of slowing. It is responsible for updating its position and speed at each time step, and turning to a new road.
3. Class `Intersection` that contains the roads and the traffic light associated with the roads.
4. Class `TrafficLight` to update the color based on the green light time.
5. Class `Simulation` to initialize everything, update the configuration of each of the above classes, and to run the simulation. At each time step, it will also add cars to the roads (if the frequency is correct), update traffic light and intersection, and update the roads.

To make sure that the code is working, I tested out the code with this configuration, with short roads, and where we don't allow slowing down, the speed can only be 0 and 1, and cars can only turn straight:

```
test_configs = {
    "Road": {"length": 5},
    "Car": {
        "max_speed": 1,
        "prob_slow": 0,
        "turn_directions": {
            Turn.STRAIGHT: 1
        }
    },
    "TrafficLight": {"green_light_time": 15}
    }
}

test_simulation = Simulation(
    roads=roads,
    intersections=[intersection],
    road_add_car_freq_dict={
            NorthEast47EW_Section1: 1,
            NorthEast47WE_Section1: 1,
            UniversityWayNS_Section1: 1,
            UniversityWaySN_Section1: 1,
        },
    new_config = test_configs
)

print(test_simulation)

test_simulation.run(10, debug=True)
```

We would want to see that:

1. The simulation initialize a car at the first time step and at every time step afterwards.
2. The green light would remains at the first road we have in the intersection, which is `NorthEast47WE_Section1`. Thus, in the 10 second time, only `NorthEast47WE_Section2` will have cars, the rest will remain empty.
3. The cars update correctly, and by timestep 10, all the cars would be congested in the other roads.

When running, the results seem to be correct. The full printed result is in the code, but I will add a small section here:

```
At 0s:
Green light at North East 47th WE - Section 1
North East 47th EW - Section 2: .....
North East 47th WE - Section 2: .....
University Way NS - Section 2: .....
University Way SN - Section 2: .....
North East 47th EW - Section 1: 1....
North East 47th WE - Section 1: 1....
University Way NS - Section 1: 1....
University Way SN - Section 1: 0....

At 6s:
Green light at North East 47th WE - Section 1
North East 47th EW - Section 2: .....
North East 47th WE - Section 2: .1...
University Way NS - Section 2: .....
University Way SN - Section 2: .....
North East 47th EW - Section 1: 0.100
North East 47th WE - Section 1: 0.1.1
University Way NS - Section 1: 0.100
University Way SN - Section 1: 0.100

At 9s:
Green light at North East 47th WE - Section 1
North East 47th EW - Section 2: .....
North East 47th WE - Section 2: 1.1.1
University Way NS - Section 2: .....
University Way SN - Section 2: .....
North East 47th EW - Section 1: 00000
North East 47th WE - Section 1: 11.1.
University Way NS - Section 1: 00000
University Way SN - Section 1: 00000

At 10s:
Green light at North East 47th WE - Section 1
North East 47th EW - Section 2: .....
North East 47th WE - Section 2: .1.1.
University Way NS - Section 2: .....
```

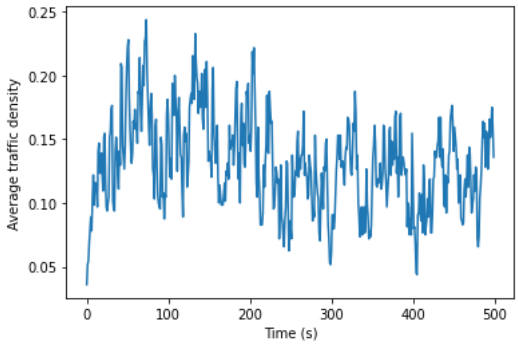
```
University Way SN - Section 2: .....
North East 47th EW - Section 1: 00000
North East 47th WE - Section 1: 0.1.1
University Way NS - Section 1: 00000
University Way SN - Section 1: 00000
```

Simulation Configuration

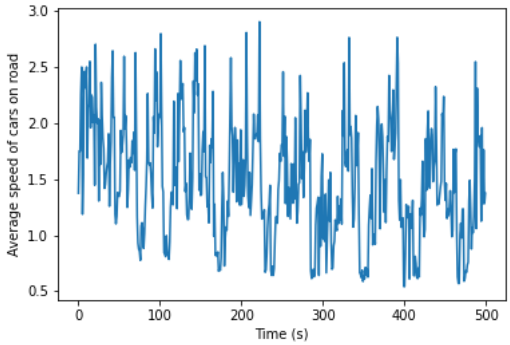
After testing the code, I want to use actual numbers based on what I observe on the street. From the empirical data, I note that:

- On average, a car would arrive every 3 seconds for the University Way Street both ways. For North East 47th street, a car would arrive every 7 seconds.
- The shorter Northeast 47th street seem to have a maximum of 20 cars, the University Way street have roughly 60 cars in total.
- Traffic light would stay green for 30 seconds at each side of the road.
- Cars have to slow down pretty often because university students seem to jay walk a lot. I updated the probability of slowing down to be 0.3.
- Cars mostly go straight (probability of 0.6), and turn left and right at equal probability (probability of 0.2).
- The maximum speed of the car stays at 5, mainly because it's a pedestrian road so cars cannot travel that fast.

I initialized it, ran it for 1 time for 500 seconds and got the following results:

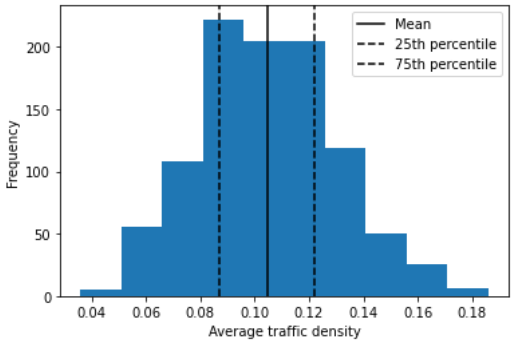


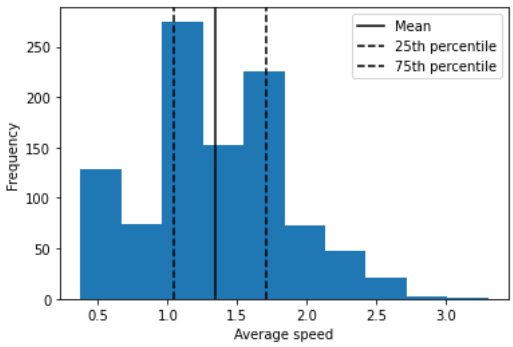
It seems that average traffic density never really surpasses 0.25 and stays mostly constant at 0.15 This seems to match our actual observation that the road is rarely, if ever, is congested. I also plotted the average speed of the cars on the road:



The values change in cycle which I'm assuming is dependent on the traffic light (because NorthEast 47th Street is probably easier to be congested than University Way Street). But we can see that the speed is roughly between 1 and 2.5.

We can now try running the simulation 1000 times to see the average result.





Seems like the traffic density hovers around 0.09 to 0.12, and average speed is between 1 and 1.7.

Theoretical Analysis

I attempted to use the mean-field approximation. While doing these theoretical analysis, I encounter these problems:

1. Taking into account the traffic light at the intersection. We can treat it as an obstacle with speed of 0 and that we will meet it with probability of 0.75 (probability of the red light) * 1/ length of the road. We add this to the probability of car in the distance, and would later need to scale the speed value to sum back to 1. Note that this doesn't take into account the possibility of other cars being slow down due to the red light, so the theoretical analysis will be more optimistic than the actual traffic density from the simulation.
2. Our calculation makes use of frequency (how many seconds before a car arrive), but we need car density. We can think of it as, in 1 second, the probability of a car being on a spot = 1/ frequency. However, this only applies to the first section (before the traffic light).
3. Even if the traffic light is green, there's a possibility that the next road would still have obstacles and the car stays in place. And since cars have different probability of turning, it makes it harder to calculate the density of the second section of the road. So the second section of the road will have a different density which is calculated based on probability of turning of the other roads.

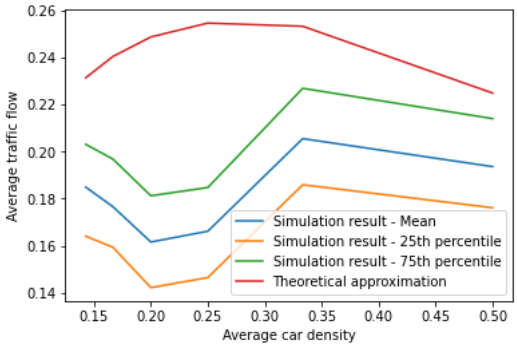
Road Section	Density	Explanation
University Way NS Section 2/ University Way SN Section 2	p = (1/3 * 0.6) + (1/7 * 0.2) + (1/7 * 0.2)	There's a 60% chance the car will go straight from University Way Section 1, which has car density of 1/3, and 20% chance that the car will be from North East 47th street either way.
North East 47th EW Section 2/ North East 47th WE Section 2	p = (1/7 * 0.6) + (1/3 * 0.2) + (1/3 * 0.2)	There's a 60% chance the car will be from North East 47th EW Section 1, which has car density of 1/7, and 20% chance that the car will be from University Way street either way.

With all of these in mind, I modified the code in Session 9 to calculate the density of each road, then calculate the average probability.

We get the mean of 0.249, which is much higher than the code that simulation of the code. This matches our expectation that the theoretical analysis will give a more optimistic result with higher average traffic.

Stress-testing the roads

What happens when we increase the density of the roads? To find out, I plotted the thereotical versus simulation results as frequency of the cars on the road increase from 1 car every 7 seconds to 1 car every 2 seconds. I then converted it to average car density and got the following plot.



The result for the theoretical approximation is as we expected, because as average car density increases, the road is less sparse, and thus the average traffic flow would increase. As it continues to increase, however, the roads become congested and the average traffic flow will decrease. It seems to peak at car density of 0.25 (i.e. 1 car every 4 seconds).

The simulation result however, is more interesting and unexpected, because it seems to dip at 0.20 and 0.25, before climbing back up and peak at 0.35 average car density. I'm not very sure if there's a good explanation for it, maybe there's a bug in my code? But it seems like on average, the road is not very congested, which is a good thing.

Closing Thoughts

To test this out further, we can try:

- Varying the probability of cars slowing down
- Varying the green light time
- Varying the probability of turning

And see what would lead to a good average traffic flow. I'm not sure if any of this is useful for urban planning purposes, but they make for a fun exploration!

Learning Outcomes

- #Modeling: I have explained my model clearly, whith clear assumptions, structure and results.
- #PythonImplementation: I have used object-oriented programming to structure a working simulation with clear visualization of state and objects, as well as testing configuration to make sure that the code functions as expectected.
- #CodeReadability: I have used accurate and clear variable names, include docstrings and comments in my code block, and followed the pep8 convention throughout the assignment.
- #TheoreticalAnalysis: I have calculated the mean-field approximation and given reasoning to my assumptions as well as pointing out flaws of these assumptions and the consequences.
- #EmpiricalAnalysis: I have explained the results obtained from the simulation clearly both visually and in prose, with clear interpretations and good choice of variable measurements. Each simulation is run 100 times and the average value and confidence interval is displayed.
- #Professionalism: I have written a clear report accompanied by a well-organized Python notebook with clear variables and units of measurements, good captions, axis labels, etc.

References

Session 9 Code Workbook. https://sle-collaboration.minervaproject.com/?url=https%3A//sle-authoring.minervaproject.com/api/v1/worksheets/0c17ad90-be56-4223-b009-b1a01a967daa/&userId=10053&name=Ha+Tran+Nguyen+Phuong&avatar=https%3A//s3.amazonaws.com/picasso.fixtures/Phu

10-
15T11%3A57%3A55.508Z&noPresence=1&readOnly=1&isInstructor=0&signature=7f3d39cde84f43c73ec7a3b51d7739286733c