Project 2: Traffic Simulation

Members

- Jas Pyneni – Event-oriented Queueing Model
- Billy Vaughn – Cellular Automata
- Napa Vananupong – Activity Scanning Model

Project Description

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

Conceptual Model

17.21min of total data

We will be simulating traffic traveling from 10th to 14th. We are only simulating traffic in one direction. We will be basing the simulations off of the entrance time and total time in the corridor. We will be analyzing the distribution of vehicle travel time in relation to levels of traffic, as measured by vehicle arrivals per minute. The three model we will be using are: Event-oriented queueing model, Cellular automata, and the Activity Scanning model.

Event-oriented Queueing Model (intersection by intersection)

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

I will eventually model this intersection by intersection, with the above three events as handlers per intersection. Each intersection will have a different priority queue. For the initial checkpoint and base model, I will model one intersection, with each car stopping at the beginning of the intersection for 2 seconds (simulating a stop sign, which will be further developed to be a traffic light) and then crossing the intersection in 20 seconds, before departing from the intersection. I will model it such that no car can enter the intersection until it

has waited for 2 seconds at the stop sign and the previous car has gotten halfway through the intersection (wait time per subsequent car = 12 seconds = 2 seconds stop sign + 10 second halfway cross). In the final model, I will simulate each lane per intersection with individual variables so that a crossing event will only happen if the intersection appropriate to that lane, is open. Based on that, I will also add functionality to switch lanes in an event if one lane is free over the other. I will generate the arrival times using random.expovariate(1/mean) to get arrival times exponentially distributed around the specified mean, two seconds. All these are assumptions to generate a base simulation model and will be updated with actual instances, such as traffic lights, turns, and intersection to intersection dependency (traffic backing up from one light to another) in the final model. This model can be run by going to the Event_Oriented_Jas directory and running "python simulationengine.py"

Cellular Automata

My model runs on a random interarrival time at the moment. In each time step, there is a chance of a new car entering determined based on the newCarChance variable at the top. New cars can also only enter when the 10th street light is green since this is the start of my model. The cars in the model are currently fixed to their lane as a simplification, but will be able to move lanes in the final simulation if there is space to move over and the car in front of them is not moving fast enough, or in a small percentage of times randomly to simulate sporadic behavior of drivers. The cars follow the following rules to drive:

1. Cars will move forward by their speed variable at each step, at the end of their step
2. If they are going less than their max speed, they will increase their speed by one
3. If there is a car in front of them prevents them from going their full speed, they will go as far as they can in that lane.
4. If a car is crossing an intersection in a step and the light at that intersection is red, the car will stop at the intersection

Currently, the intersection light timing is also random. The lights all start as red and wait between 10 and 50 seconds to change to green. This timing and the car arrival time will also be adjusted to be in line with the provided dataset in the final model.

Activity Scanning Model

My model will use an event oriented implementation approach with an activity scanning model world view. With this approach, the simulation concentrates on the activities and especially the conditions that allow an activity to begin. At each clock advance, the conditions for each activity are checked and only if the conditions are true, will the corresponding next even occur. As the model is event driven, the simulation time will be continuous and state variables will be updated when an event has occurred. Events occur at irregular points in simulation time and simulation time advances from one event to the next. I will create a simulation application which contains the state variables, code modeling system behavior, and I/O and user interface software. This application will make calls to the simulation engine, which is calls to schedule events, another file in which the event list is managed and the management advances with simulation time. This file will call the simulation application as an event handler. I will also make files for each object, such as the car, the lights, and the lanes (by the final submission) to keep track of their states. Eventually I will find a random number generator. There are three main events my model will use:

1. the event that a car has arrived at the traffic light. The condition is, if the light turns green, the car will move forward onto the next event (going through the road), if the road is not at full capacity (the lane). Otherwise, the car must stay there till the light turns green.

2. The second event is that the car has entered the lane and is traversing through. If there is space available at the end it may run through (var boolean isLaneFree = T/F).
3. The last event is the departure event, where the car departs the lane and disappears out of peachtree 14th. By the final checkout I will make more than one lane and have them able to switch lanes.

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