CX 4230 Project 2 Final Report

Team Lawfael/Rawlence

The Problem:

In the event of a campus wide evacuation, having an estimate of the overall evacuation time and the factors involved in the process would be exceptionally useful for campus officials to effectively handle the crisis. Our model was built to simulate such an event under a variety of circumstances. In our simulation, a dangerous event occurs on the west side of campus, causing a campus wide evacuation towards I-85. The results of the simulation provide an estimate for the speed in which campus can be evacuated and how could traffic direction policies affect the final evacuation time

The Model:

Our conceptual model focuses on the three most important factors in an evacuation scenario: the parking lots, roads, and traffic stops. We will assume that no cars enter the campus during the crisis and that the amount of time it takes for individuals to reach their cars is negligible compared to the overall amount of time needed to exit campus. We also assume that no pedestrians or accidents interfere with travel on the road. Thus, the simulation only depends on how quickly a certain number cars in the parking lots around campus can exit campus based on the road size and direction taken at traffic lights.

Each road was considered as a first-in-first-out (FIFO) queue, with the capacity directly related to the length of the road. This length was estimated using Google maps. Further, the speed of the cars, and thus the time in which cars remain in the queue, is inversely related to the number of cars in the queue. Thus, if only one car is driving down Hemphill, that car can go the speed limit of 30, but given a large amount of traffic, the car must go slower. There are a total of seventeen parking lots spread throughout campus, and there are three exits onto I-75. All other traffic, such as in midtown, is ignored.

To simulate a real life scenario, a random number of cars will be generated in the parking lots according to a uniform distribution. The choice of this distribution was to simulate a common day at Georgia Tech, where all the cars are evenly spread. The rate at which cars leave the parking lot and enter the road is linearly proportional to size and occupancy of the road ahead as well as the number of cars in the parking lot.

To coordinate all the processes that occur in the model, we employed both a local clock at each intersection as well as an overall global clock. Each intersection has its own local clock that keeps track of when the next car can move through. The global clock is used to synchronize the movement in all the intersections; at each tick, each local clock updates.

Accurately modeling the behavior at stop lights is slightly more nuanced. For each road entering an intersection, the front car generates a work request, which dictates when the car leaves and which road it enters. The speed in which these processes occur depends on how full both the entering and exiting road is. Further, the choice of which road to enter depends on the way traffic at intersections is handled. In our model, there are two possible situations: people are left to decide which direction they wish to go, or an officer is present to guide traffic. We broke these different scenarios into two types of behavior.

The first traffic light behavior used what we called a greedy heuristic. In the event that no officers are present, one can assume every individual will always choose to go East. Further, if the road east is full, the individual will randomly choose another available road either in the North or South direction. This "greedy" approach accurately models real life scenarios, in which where people tend to bunch together in large traffic jams in the attempt to take the best possible road.

The second traffic light behavior takes into account the presence of traffic officials. We called this the police officer heuristic. In this scenario, the officers send traffic down the least crowded road available. Naturally, if no road is available, then traffic comes to a stop. This roughly resembles the way traffic officials actually operate. So, running the simulation with both the greedy and police officer heuristic allows us to observe how intersection flow impacts the overall evacuation time.

Code:

We chose not to use the SimPy library for several reason. Mainly, building our own software from the ground up gave us much more control over all variables involved in the simulation and also allowed us to implement whatever software architecture approach we found was most intuitive. Another factor was the overall lack of documentation on SimPy. Thus, no external modeling libraries were used.

Regarding the overall structure, our software consisted of one main class (*Traffic.py*), along with a class with helper methods (*Node.py*) and a file with the map data (*GTData.csv*). The main class *Traffic.py* consisted of the general simulation scenario code, along with the Police and Greedy heuristics. This class also contains a series of methods which run the simulation under a variety of circumstances.

The helper file *Node.py* contained all the helper classes and methods necessary to represent the parts of the model. Its main function is CreateMap, which creates adjacency lists based on an inputted map. Those adjacency lists consist of Node and Edge classes in the format (Node: Edges). The Node class is used to represent intersections. The Node class also contains a heap, which was used to store the work requests, which were discussed in the conceptual model. The Parking lot class, which is a subclass of the Node class, contains all the extra information to represent a parking lot. The Edge class is used to represent road between two nodes and contains a queue to represent capacity. The CreateMap function returns a list of all the parking lots, an adjacency list of all incoming paths, an adjacency list of all reversed paths and a list with all the possible Edges.

All of the classes in *Node.py* are used in the main simulation class to represent how the model evolves over time. Initially, the data file is processed and two adjacency lists, one for cars entering the intersection, another for the cars exiting, are created. Cars are initially generated in the parking lots and the roads are assumed to be initially empty. This makes sense given our assumptions, as in real life the amount of cars parked in the streets or already moving would be minimal compared to the scale of a campus wide evacuation. The Simulate class then takes these adjacency lists as parameters along with the desired clock tick time and the desired heuristic (Officer vs. Greedy). After the adjacency lists are processed, we proceed to process every intersection by passing them into the desired heuristic method. The heuristic then proceed to create several *WorkRequest* objects based on the roads coming in and out. A *WorkRequest* object contains the direction of the movement of a car and time needed to move through an intersection. This direction is dictated by the Police Officer and Greedy heuristics. These *WorkRequest* objects then are placed in the heap of the intersection and are sorted based on the time to execute. The time of each of work request is update every clock tick to replicate the passage of time. Whenever a Work Request's time reaches zero, the Request is deleted from the heap and a car is dequeued from the previous road and enqueued onto the next road. When cars are moved to exit Nodes, they have then successfully left the system. The simulation is only over when all the cars have left campus.

It is worth mentioning that the approach mentioned in the above paragraph reflects a synchronous model; time is updated globally among all WorkRequests on every nodes though the UpdateWorkRequests method, which is called at the beginning of each iteration. We chose to do so from an implementation perspective because it gave us a higher level of control over the simulation and highly reduces the amount of bugs. But even from a realism standpoint, traffic does become essentially synchronous in a highly congested area. Regardless, whatever accuracy may be lost by this approach is minimal, since the time units inside a Node class consist of fractions of seconds.

The data file GTMap.csv consisted of information on all the intersections on campus in the format (Incoming Street/ Next Street/Distance/Direction). This file is made by hand using Google maps. Using this data formatting proved itself to be very useful for our algorithms, since all we need to know to redirect traffic is where the intersections are, their directions and how long the roads are. The latter information was also useful in calculating the maximum road capacity, as we considered the average car length to be 4.5 meters.

Random Number Generator

Random number generators play a central role in the simulation. For example, when the map is generated, a random number generator is used to assign the number of cars to each parking lot as well as dictate some of the decisions made at the intersections. The idea behind using random numbers instead of a fixed value is simple: a model of the evacuation time of campus should be independent of a particular day's traffic and take into account some variability. Otherwise, an unintended bias is introduced. Thus, it's important to be sure that the random generator being used is indeed "random", at least in terms of its general behavior.

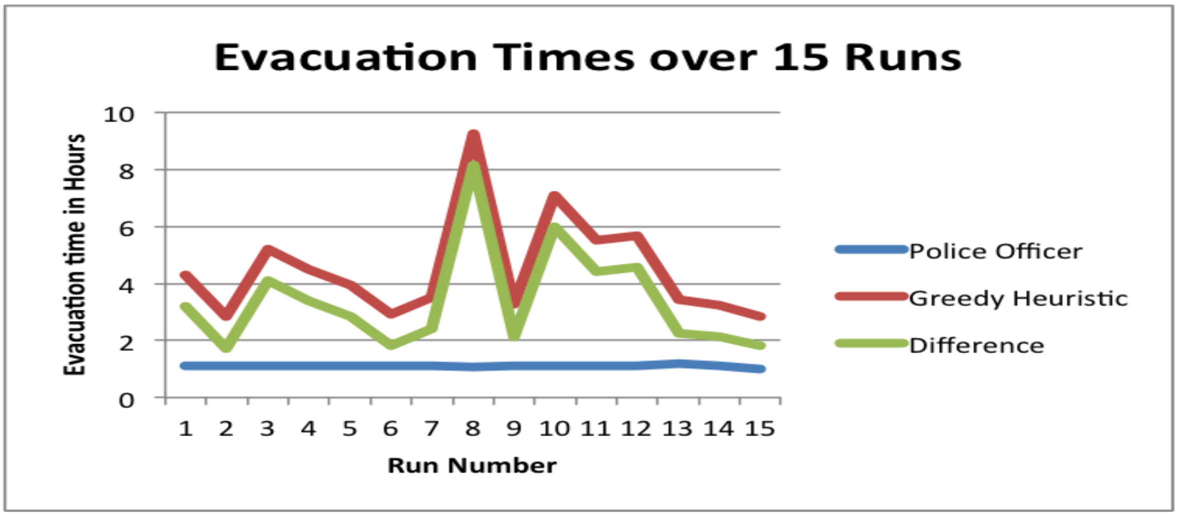
To test the random number generator, the chi square test was employed. Various sample sizes were used, and the resulting p-value and chi square statistic are shown in the table below.

Number Of Samples	p - value	chi square statistic
10	3.86E-27	147E+02
100	0	2085.411255
1000	0	16716.03192
10000	0	168250.7207
100000	0	1693203.145

From above, we can observe that as the number of samples increases, the p value actually goes to zero. This indicates the probability of the null hypothesis being wrong is essentially zero. In this particular case, the null hypothesis is that every point is uniformly distributed. Thus, according to the test, the random number generator appears to be random. Considering this is the default random number generator of python, this is relatively unsurprising. Still, for the experiments performed later in the report, we have a firm foundation to be believe that each run is indeed independent of the other, and that random clusters do not form in the map due to unintentional random number generators.

Results and Analysis

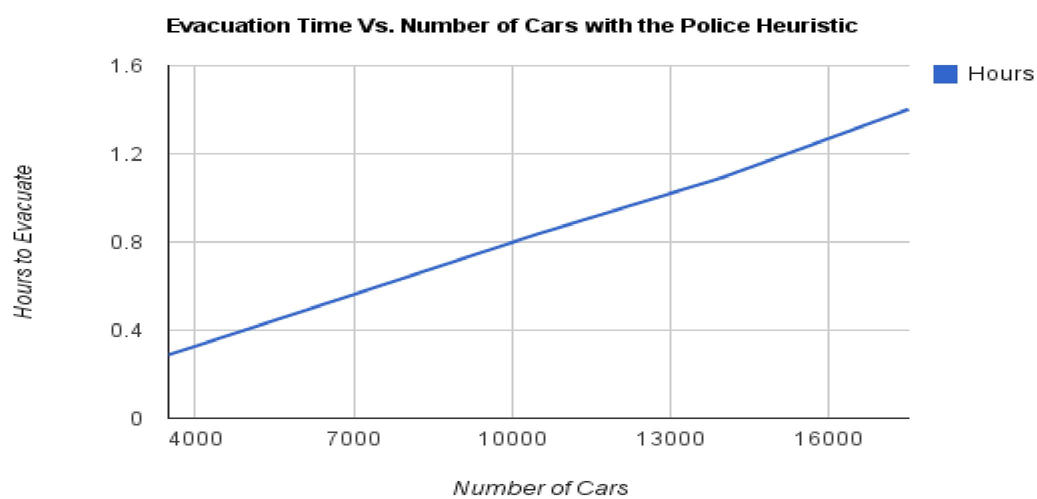
The first experiment was to run the simulation with approximately 14,000 cars on campus with both the greedy heuristic and officer heuristic. Fourteen thousand was chosen based on the number of parking lots seen on the campus and their corresponding size. Given the random nature of the simulation, each heuristic was run a various number of times, starting at five, then fifteen, then thirty, and finally fifty. This not only gave us a better picture of the true evacuation time according to the model, but also provided us with enough data to formally our estimate confidence in the results. The important results are shown below.



Number of Runs	Police Average in Hours	90 % Confidence Interval for Police in Hours	Greedy Average in Hours	90 % Confidence Interval for Greedy in Hours	90% Confidence Interval of Difference
5	1.110	0.006	3.71	0.61	0.613
15	1.104	0.004	4.85	1.00	1.06
30	1.107	0.003	5.03	0.62	0.619
50	1.107	0.002	4.84	0.57	0.266

Several trends can be seen in the data above. Perhaps the most striking is how consistently the police heuristic performs. Even with only five runs, we can predict the evacuation time to be around 1.1 hours with 90% confidence. Yet, this is not terribly surprising given the nature of the heuristic. The officers try to evenly distribute traffic throughout the roads, and given enough of a traffic jam, the exists become the limiting factor. This causes the cars to end up being dispersed in the roughly the same way in every simulation. On the other hand, the greedy heuristic performs much worse and with much less predictability. On average, it takes almost four hours longer than the police heuristic. The source of this extra delay can easily be seen in the command line visualizer. Because all cars frantically try to go east, a few roads become the bottlenecks, which slows down the simulation significantly. The large confidence interval further confirms the variability of this heuristic, since the range of values can be anywhere in a 1.2 hour time span given 90% accuracy. One possible reason for this is that after watching the visualizer, it seemed that the distribution of cars in the parking lots is paramount to how quickly the cars leave. If the cars are spread out to begin with, it allows traffic to flow much more smoothly than if they start out congested in the first place.

Another thing to consider is how the overall number of cars affects evacuation time. Below is a graph showing this effect.



As shown above, there seems to be a linear relationship between the number of cars and the time it takes to evacuate. This follows logically given the relatively small capacity of the exists. Once the exists become full, the rate at which cars exist determines the evacuation speed. Thus, given enough cars to fill the exists, the evacuation time becomes linear with the time of exit and the number of cars.

Discussion and Conclusion

Overall, the model provided a few interesting insights into how traffic flows. Namely, distributing the number of cars evenly provides a large and consistent speed up compared to a more simple of approach of just trying to go east. While this is very intuitive, it directly explains why panic situations tend to be much slower; people don't think as rationally as one might expect, and this causes natural build ups. This trend also shows why the presence of a governing force is beneficial. While only one coordinated effort for handling intersections was explored in this report, more nuanced approaches could be used. For instance, the queueing strategy could follow the max-flow algorithm, in which people go in the direction of the overall least crowded path to an exit, as opposed to just the least crowded nearest road. Another approach could be for officers to force traffic to follow a certain preordained traffic pattern, as opposed to dynamically optimizing like in the situation above. In addition, making certain roads two way while never using others might provide a speed boost as well.

Another aspect of the model that clearly could be refined is how the simulation governs the time of intersections and work requests. Clearly, an hour to evacuate fifteen thousand cars would be a spectacular feat of coordination and most likely isn't possible in a real life panic scenario. Our model most likely underestimates for several reasons. The first is that all relationships are linear in the model, including the number of cars to the speed of departure. While this made implementation relatively straightforward, one has a sense that there's a more exponential flavor to this trend: in a real crisis situation, things become disproportionately slow, even if the number of cars themselves isn't much more than a normal day. Another problem may be simply the constants we used. For instance, we estimated that the exit time onto the highway is 40 seconds, even when the road leading to the exit is completely gridlocked. Anyone who has driven in Atlanta before knows this is being generous. However, in order to accurately model this, traffic from neighboring areas like Midtown and even the highway itself would need to be considered, as these are prominent factors of departure time despite being ignored in our simulation.

One last way to improve the accuracy of the model would be to analyze large amounts of traffic data on campus, perhaps taken from GPS readings. Instead of using somewhat arbitrary constants that seem accurate based on observation, one could apply machine learning algorithms to more effectively measure the time taken at each intersection during various times of the day and different levels of traffic. This would allow the heuristics to make the most informed decisions possible.