Ph11 First Hurdle

Research:

My first step for approaching this problem was to do some general research. Utilizing Google Scholar, and the Caltech WoK, I found articles pertaining to vehicular traffic modeling.

I have included all of the articles I referenced in this folder. (All except #13, due to its length – 170 pages.) They are numbered, and I shall refer to them by number throughout this paper.

Reading #13. I got a good overview of the various methods used to model traffic flow. The most common forms are fluid dynamic models, gas kinematic, follow the leader, and cellular automata.

Given the nature of this problem, I felt a cellular automata approach would be most fitting. The question specifies some starting conditions for a specific car. In fluid dynamic and gas kinematic models, individual cars are not distinguishable.

Also, as intriguing as the fluid dynamics approach seemed, I will be the first to admit my lack of expertise with differential equations. Following this route would have been difficult for me.

Left with follow the leader, and cellular automata models, I choose the later, partly because literature seemed abundant, partly because I could envision its implementation, and partly because the literature suggested it was quicker than follow the leader models.

I found document #1 to be especially helpful. It laid out clearly and simply a simple cellular automata model. After reading further, I would discover that this NaSch model (created by Nagel and Schreckenberg) would serve as the standard for all CA (cellular automata) models.

Data Collection:

The next task to completing this hurdle was to collect ample amounts of data on the road I was dealing with. The I-84 stretch of highway from Hartford, CT to Danbury, CT.

I poked around on the internet a bit. Didn't find anything all together promising. So, I decided to call the Connecticut Department of Transportation to see if they could help me get some information. I was transferred to the engineering department, and a nice fellow named Al Iallonardo. He said he could help me out if I faxed him what I was looking for. I did (fax is attached as doc 8). Al pointed me to the Connecticut Highway Log. A collection of data for all of Connecticut's highways. I printed off the part pertaining to our stretch of highway (doc 11). The Highway Log includes all on ramps, off ramps, lane changes, intersections, crossings, and town line crossings. A very extensive source of information. I also asked Al about speed limit changes along the route. He told me that he was pretty sure it was 55 mph for the majority of the journey,
but he would have someone check on it and get back to me. Monday morning, I received a call from the Connecticut Department of Transportation, a kind fellow had gathered the information I needed. Apparently the route in question has about 30 or so speed limit changes, everywhere between 35 and 65 mph. He also assured me that it was for a large part of the journey 55 mph, but said that he would mail me the information. I should be receiving data for the speed limits in the coming days.

**Programming Part I:**

Having more than enough data than I knew what to do with, I began to consider the actual implementation of this problem. I have to admit that I am not an accomplished programmer. I took a semester of C at UCF a couple summers ago. Taught myself some Python, and am taking CS1 currently. In the end, I decided to grapple this problem in Python because I find it easiest to program, very easy to do something interesting with the data, and I felt fairly comfortable with it.

Next I turned back to my documents, particularly document 1. I have included a copy of my first go at this problem in document 9. In my first model, I created a two dimensional array, each column signifying a car, with a position and velocity. Utilizing the rules laid out in the NaSch model, I created steps for acceleration, deceleration, random deceleration, and movement, taking car to apply each rule on a parallel array before copying the contents of this dummy array back into my core one. Naturally, my first go was dealing only in with a single lane highway.

I modified the NaSch model slightly. Knowing that I would need to be able to express units of 55 mph, and a 10mph difference specifically. I took the strategy of some of the later NaSch models by designating each space unit as less than a car length. In my model: each car takes up 4 spaces. Since a car in a jam takes up about 25 ft, each unit of my model corresponds to 25/4 ft. As such, 55 mph corresponds to a vmax of 13, and a velocity of 45 mph would be 11. Therefore, I am able to describe the entire highway in terms of 63800 space units, which with cars traveling at 55 mph with safe following distances, makes 3752 cars. Using my method, I was able to preserve each unit of time as analogous to 1 second.

One more point of note: The NaSch model has cars decelerating by one unit with some probability p. From reading the literature it seemed that the p that led to the most realistic behavior in a single lane NaSch model was a p of 0.1. I know from reading #13, that I should be able to correct this p to be most accurate by comparing my models flux and flow vs density graphs with actual observed data. Unfortunately, due to time constraints and my lack of knowledge in this area, I had to trust the validity of a p of 0.1

Another point of note:. At first I ran the simulation such that when a car reached the end of the highway, it would stop being emulated. This helped out a lot with computation time, but after looking at the results, I noticed this would lead to a backward front starting from the finish point of undisturbed traffic. The reason being that traffic within a certain distance of the finish in terms of time and space would have no cars in front of it to create traffic disturbances or jams. Naturally, this is not like the real world. In the real world, traffic continues beyond Danbury, CT. So, to correct, I realized I needed to emulate some
traffic beyond the finish point. I decided to instead treat my stretch of highway as a closed loop, with no end points. This way, I could have traffic disturbances right up to Danbury, without emulating points that I could not observe easily (allowing me to check for irregularities). The only problem with this approach is that it is possible for a traffic jam that occurs early on the highway to affect our noted car both at the beginning and end of its journey. However, due to the very obtuse nature of my graph (much wider than it is tall), this situation is rather rare, and I do not think it would adulterate the model much.

I have included a printout of the simulation running. Each dot represents a car. Position runs alone the x axis, left to right. Time runs along the y axis, top to bottom. The red dot marks our tracer car.

**Programming Part II**

Having come up with what I felt was a satisfactory model for modeling traffic in a single lane highway, I thought I would try and expand my model to multiple lanes. Looking at the Highway Log, it appears as though, there are much too many lane changes to implement them all. Doing so would require much more computation time and power than I have at my disposal. (I was pressed for time as it was).

I thought the conversion from one lane to multiple lanes would be pretty straight forward. Unfortunately for me, the way I emulated one lane (with a two dimensional array of cars), made it difficult to imagine adding additional lanes without drastically affecting computation. So, I basically had to rewrite my entire model, instead creating arrays (and their dummies to allow for parallel updating) which would correspond to the road, with each cell being a unit of space. If filled with a 0, that point in space would be unoccupied by a car. If filled with a number > 0, that would signify a car. I would then have an array (and its accompanying dummy array) which would have one cell for every car. Indexed by their number (present in the street), each car would only contain information about its velocity. By doing so, I was able to keep track of which cars were in front of which (by looking at the road array), as well as updating the velocities of my cars in a simple manner.

This is how I spent all of my Sunday night. Trying to get this model to work. And alas, it does... to a point. Running the simulation, it appears as though it works, but after a certain period of time, my cars all seem to slow down to a velocity of zero. I believe this may be due to how I am calling my randomization procedure, but unfortunately, I do not have time to work this out.

I have however, included a printout of one running of this two lane model. I assumed two lanes for the duration of the trip (to aid in my computation). Green represents a car in one lane, Blue in the other lane. Teal means there is a car in both lanes at that point, and as always, the red dot is our tracer car.

**Results**

So, running my one lane model tells me a few things about this trip.

1.) The random deceleration of probability $p = 0.1$ leads to a lot more traffic jams than our single preconditioned one.
2.) The starting condition of slowing a single car (green in the model) by 10 mph for 10 seconds, while leading to a jam, is not more significant than any of the other jams along the way.

3.) This tells us that the time it takes to complete the trip is highly variable according to how the jams develop and disperse.

Unfortunately, due to time again, I was not able to run the model for the other conditions (1 mph for 1 second, and 5 mph for 5 seconds). But I can tell from my experience that I would not expect the 1 mph decrease to have any significant affect on travel time, as it's influence would be flooded out by other random decelerations. Secondly, the 10 mph delay for 10 seconds does little to affect travel time, so the 5 mph for 5 seconds would do even less.

In effect, I would not feel at all bad about reporting the same time for all three trips, which consequently for my model, the trip is estimated to take:

My model took 1498 seconds to run for 7000 iterations, or 7000 seconds or highway traffic along the I-84

Possible Improvements

Naturally, a great improvement would be to see my multiple lane model working. I could then add increasing complex conditions (on ramps, off ramps, individual lane changes, speed changes), each of which giving another level of reality, while compromising hugely computation time.

Also, I could have implemented the brake light addition seen in many of the newer NaSch models, especially the late 90s and early 2000s. However, seeing as I lacked the time, I could not really implement this condition.

Conclusion

Traffic modeling is terribly difficult, but the problem was terribly fun.

Additional References

#13
P89. D. Chowdhury, L. Santen and A. Schadschneider, "Statistical physics of vehicular traffic and some related systems.