Highway Network and Traffic Simulation

Haoting Li, Chenwei Yue, Qi Zheng, Junlin Liu

1 Project Description

Traffic models are fundamental resources in the management of road network, and traffic flow modeling is an important step in the design and control of transportation systems. Traffic congestion is a problem annoying all the drivers. Establishing a model to simulate a real traffic and find the reason for congestion is meaningful to solve congestion problem. A progress in the study of traffic has obtained with the introduction of the models based on cellular automata (CA)[1].

A cellular automaton is a collection of cells on a grid of specified shape that evolves a number of discrete time steps according to a set of local rules based on the states of neighboring cells. A set of simple rules can be used to simulate a complex behavior; thus, these models are conceptually simple. A cellular automaton consists of a regular uniform lattice[2], usually finite in extent, with discrete variables occupying the various sites. The state of a cellular automaton is completely specified by the values of the variables at each site. Cellular automata models are capable of representing individual vehicle interactions and relating these interactions to traffic flow. By allowing different vehicles to possess different driving behaviors like acceleration and lane change, CA models can more adequately capture the complexity of real traffic [3].

One simple traffic model is defined as a one-dimensional array with L cells with closed (periodic) boundary conditions. This means that the total number of vehicles N in the system is maintained constant. Each cell (site) may be occupied by one vehicle, or it may be empty. Each cell corresponds to a road segment with a length l equal to the average headway in a traffic jam. Traffic density is given by $\rho = N/L$. Each vehicle can have a velocity from 0 to vmax. The velocity corresponds to the number of sites that a vehicle advances in one iteration. The movement of vehicles through the cells is determined by a set of updating rules[4]. These rules are applied in a parallel fashion to each vehicle at each iteration. The length of an iteration can be arbitrarily chosen to reflect the desired level of simulation detail. The choice of a sufficiently small iteration interval can thus be used to approximate a continuous time system. The state of of the system at an iteration is determined by the distribution of vehicles among the cells and the speed of each vehicle in each cell. We use the following notation[5] to characterize each system state.

x(i): position of the ith vehicle
v(i): speed of ith vehicle
g(i): gap between the ith and the (i+1)th vehicle (i.e., vehicle immediately ahead) and is given by $g(i) = x(i + 1) - x(i) - 1$.

In this project, a model will be built to simulate the traffic flow of several highways (e.g., I-75 highway) in Atlanta. Rules of the vehicles will be defined. Special cases like random car accident and road construction that will affect the result of simulation will also be taken into consideration. Finally, we will analyze the relationship between road condition and traffic congestion. We will try to find out what parameters (e.g., speed limit, maximal vehicle number)

can avoid traffic jam during rush hours or when accidents occur.

2 Conceptual Model

Input:

- Geographical location information of Atlanta highways
- Parameters of some highways in Atlanta (e.g., I-75) including the number of lanes, the position of exits and entrances
- Parameters of vehicles (e.g., speed, destination) running on the road.
- The probabilities that events like road construction and car accident will happen

Output:

- The time for simulation
- The visualized dynamic traffic condition
- Traffic condition indicators (e.g., traffic density, average waiting time of each vehicle) at different time periods (e.g., morning, noon, rush hour)

Entities:

- Consumer: Vehicles (Cells in a CA model)
- Resource + Queue: 2-lane highway
- Group: Intersections where vehicles might turn left, turn right or go straight

Attributes:

- **Vehicles:**
  ID: the label of one vehicle
  Speed: current speed
  Maximum velocity: the maximal velocity allowed by the vehicle itself
  pRight: the probability that this vehicle will turn right at one intersection
  pLeft: the probability that this vehicle will turn left at one intersection
  Position: the vehicle's current position in the lane
- **Highway:**
  Maximum speed: the maximal speed for all vehicles in this road
  Lane availability: true if the number of vehicles in this lane <= limit
  count: number of vehicles of this lane

Event:

- **Vehicles:**
  Stop
  Change lane
  Enter
  Exit
  Turn left
  Turn right
- **HighWay:**
  Change maximal speed
  Accident: an accident occurs, and the road condition of this highway will change

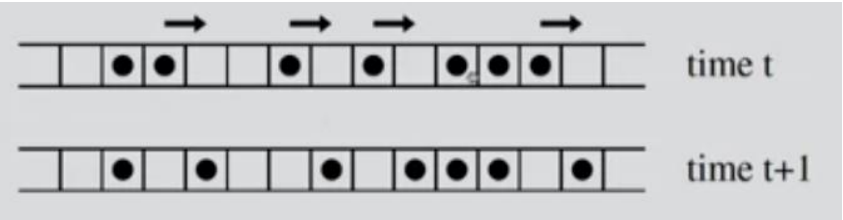
Activity:

- Vehicles:
  Run
  Accelerate
  Decelerate
  Stay

Assumptions and Rules:

1. Basic Rule. The movement of vehicles in a single lane of a highway follows Wolfram rule 184[6]: All vehicles in a single lane move in a single direction. A vehicle can move only when the downstream cell is free.

For example, the graph below shows the traffic flow at time t and time t + 1. The black dots represent the vehicles in a single line.



2. Vehicles acceleration with large front open space: If the velocity of vehicle i, i.e., $v(i) < vmax$ and gap in front of the vehicle i: $g(i) \geq v(i) + 1$, then $v(i) = v(i) + 1$.

3. Slowing down due to the blocking of other vehicles ahead: if $v(i) > g(i) - 1$, then $v(i) = g(i)$.

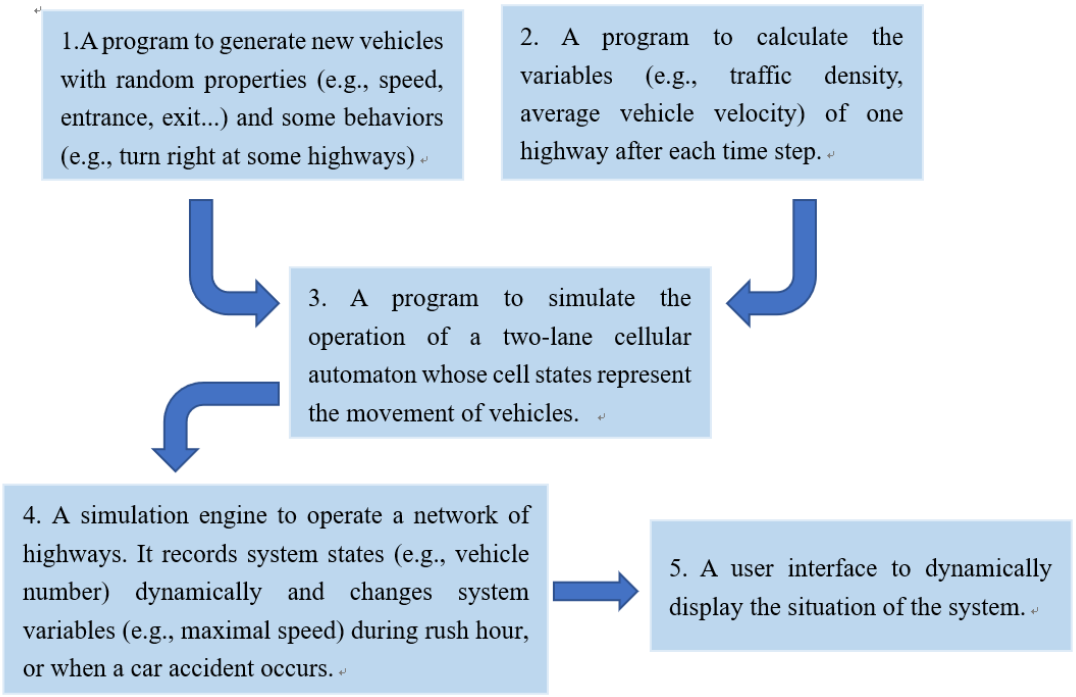
4. Vehicle motion: the vehicle is advanced v(i) sites.

5. During lane changeover, only transversal movement are allowed (i.e., vehicles do not advance)

6. Once a vehicle changes lane, it remains in that lane until it becomes more desirable to move back to the other lane (i.e., vehicles do not always return to the right lane).

7. Once all lane changes are made, the updating rules from the single lane model are applied independently to each lane[7].

3 Block Diagram of the Model



4 Data Collection

Interstate 75 (I-75) is a major interstate highway in the Great lakes and southeastern regions of the United States. The whole length of the highway is 1786.47 miles, which is 2875.04 km. Within Georgia, I-75 is 355.11 miles (571.49 km) in total. I-75 runs concurrently with I-85 due north over the Downtown Connector through the central business district of Atlanta. The areas where I-85 and I-75 run concurrently are some of the most traffic-prone streets in the nation. There are 161 different positions where exits are set along I-75 in Georgia State. Some of them locate really near each other and some others also have multiple exits in the same position. The speed limit on I-75 is typically 70 MPH with the exception of a few areas where it drops to 55 MPH. However, it appears that when you enter the state of Georgia, the speed limit changes constantly, i.e. from 70 to 60, then to 55, and to 65. To simplify the software in order to make it runnable, we assume that there is an exit every 5 kilometers.The real data will be added to the software in the next stage.

5 Software Description

5.1 Overall Structure

This program simulates and the traffic flow of several highways (e.g., I-75 highway) in Atlanta. We will also build an UI to show how traffic flow changes during one day.

The movement of vehicles on the highways is based on some rules (details are given in the description of “MultiLane”).

The classes of this software are shown below.

5.2 Vehicle Class

This class defines several important properties of one vehicle:

1) type

There are three types of vehicles. Their max speed is 25m/s, 30m/s, 35m/s, respectively. The probability of the three types is: 20%, 50%, 30%.

2) speed: current speed of this vehicle

3) vMax: maximum speed allowed by this vehicle

4) probRight: the probability to turn to the right lane if it's allowed

5) probLeft: the probability to turn to the left lane if it's allowed

6) the probability to turn to another highway if it's in the rightmost lane

5.3 Lane Class

This class defines several properties and functions of one single lane of highway.

The important variables are shown below.

1) cells: an array of all the "cell"s of one lane. The length of one cell is 5 meters, so the number of cells depends on the length of each lane.

2) vMax: maximal speed for all vehicles in this lane

3) density: total number of vehicles / total number of cells. It describes the traffic condition of one lane.

4) probExit: at the conjunction of two highways, there is either an exit or an entry, the probExit is the percentage of vehicles that will leave the highway at that exit. It basically a representation of the exit frequency of passing vehicle at each highway exit. Note that it only applies to the rightmost lane.

5) probEnter: similarly, each entrance to the highway also has its entering probability.

This class has several functions:

1) addCar(): add a new vehicle into a given position of one lane

2) removeCar(): remove one vehicle at the exit

3) getPara(): return some system status of this lane (e.g., density, occurrence of accidents)

5.4 Cell class

To model the real highway network, we also defined a cell class which stores the geological information of each grid on the highway. It has three basic attributes:

1) gridSize: the grid size of each cell

2) xCoord: x-coordinate of the cell

3) yCoord: y-coordinate of the cell

5.5 MultiLane class

This class basically assembles all the lanes on one side of the highway, hence represents the channels for vehicles and defines how one side of the highway (single direction) operates.

The important variables are shown below:

1) length: The length of the highway. In our checkpoint programs and models, to avoid the case when the density of points (vehicles) are too high, such that neighboring points are clustered together, which makes it difficult to visualize the simulation and results, we assumed that the length of the road is 30km. Refinement will be done when a better UI is developed.

2) nLanes: Number of lanes on the road. One direction is applied this time. a two-direction road will be simulated in the next stage.

The class has a set of key functions to operate the highway. These functions are utilized, at each timestamp, to check irs current condition (and the surrounding) to determine what to do to update the condition. All the rules have already been described in the "Assumptions and Rules" part of Section 2 Conceptual Model in the previous texts. These functions are:

1) updateSpeed(): function to check the current state and update the speed of the vehicles on the highway based on the speed update rules.

2) updatePosition(): function to "move" the vehicle on the highway based on the position change rules.

3) checkChangeLaneLeft(): Check if a vehicle will change to the left lane, move the vehicle if it yes. Note that vehicles on the leftmost lane cannot change to left lanes. In additional, even when changing lanes is allowed, we still want to use a pre-defined probability for that lane to account for the case the driver does not want to change lanes.

4) checkChangeLaneRight(): Same as checkChangeLaneLeft(), check if each vehicle need to change lanes to the right, update the cell states if yes.

5) enterAtStart(): Generate vehicles at the start point of the highway.

6) exitAtEnd(): remove the vehicles when it reaches the end point of the highway or goes beyond the boundary of modeled section

7) checkExit: At the checking time, if a vehicle reaches the highway exit point, we first determine whether it will exit the current highway and drives to the other highway based on the predefined exit probability. If it will leave, then we check the connecting point at junction (the cell on the rightmost lane of the new highway the vehicle will enter), if the entrance point is occupied or if there are any other vehicles queuing up (in the queue), waiting to merge to the new lane, we reduce the vehicle speed, remove it from the current lane, then instead of placing it on the new lane cell which is occupied, we push it into a FIFO queue and wait. Merging/incoming vehicles must yield to vehicles that are already on the lane.

Also within this function, at each time stampe, we check if there are any vehicles in the FIFO queue at each exit point waiting to enter new highways, and if the entrance is open (not occupied), if yes, we pop out the front vehicle and place it on the new lane.Actually this method also handles the entering case, since exit from the old highway is equivalent to enter a new highway at the corresponding point.

5.6 HighWay class

This class basically assembles/combines the two MultiLane classes of opposite directions and assigns the geographic data to the highway to model the real cases.

6 UI

We use Tkinter in python2 and matplotlib library to implement our UI. Currently the UI only contains the animation that shows the traffic flow on I-75. X axis represents the length of the road we simulate, and Y direction shows the lane number. The nodes within the figure represent the cars running on the highway. Next step we will add parameter tuning functions on the UI, and also provide the user with animation play and stop control.
To make the UI more fluent and smooth for usage, we use a multithread strategy here. There are two threads in our software. One is the work thread. This thread computes the result of each step and process the data for the UI to draw the frame. There is a message queue in our code. This queue is multi-thread safe queue. Each time the data is ready for UI to draw, it will add a message in the queue. UI thread detects messages in the queue. Every time it pops only one message out and draw the frame. Therefore, the UI will not freeze during data processing.

7 Future Work

Till this point, the preliminary simulation software has been developed. All fundamental components of the model have been completed. Key utility functions have been finined. A basic simulation engine has been developed to operate a 3-lane two-way highway with exit lanes on both sizes has been developed. Basic system integration has also been performed to assemble all the components into an integrated model. The model has been tested on simplified scenarios and proved working properly. We have also done with digitizing a section of a real highway network map and assembling the data into a graph in Python. We are now ready to move forward to applying the real data on the model, simulating the real traffic, and tuning the parameters to perform sensitivity study. For the remaining time, we propose to:

1) Combine all the developed components to a complete highway network with nested multilane highways with exits and merges;

2) Apply the real geographic data to our model representation, process and analyze the real traffic data to generate input data for the simulation; The model (and the program) will be further refined by adding a few more attributes, and allowing transient/real-time data;

3) Improve the UI to better present the highway network, real-time traffic condition, and simulation results.

4) Final software integration, and final report

References

[1] Centro de Matem´atica, Faculdade de Ciˆencias, Universidade do Porto, A multi-lane traffic simulation model via continuous cellular automata, Arxiv: 1302.0488v1 [cs.MA] 3 Feb 2013

[2] Saifallah Benjaafar, Kevin Dooley, Wibowo Setyawan, Cellular Automata for Traffic Flow Modelingm, University of Minnesota

[3] Nagel, Kai, and Michael Schreckenberg. "A cellular automaton model for freeway traffic."

Journal de physique I 2.12 (1992): 2221-2229.

[4] Nagel, Kai, et al. "Two-lane traffic rules for cellular automata: A systematic approach." Physical Review E 58.2 (1998): 1425.

[5] Maerivoet, Sven, and Bart De Moor. "Cellular automata models of road traffic." Physics Reports 419.1 (2005): 1-64.

[6] Rickert, Marcus, et al. "Two lane traffic simulations using cellular automata." Physica A: Statistical Mechanics and its Applications 231.4 (1996): 534-550.

[7] Kerner, Boris S., Sergey L. Klenov, and Dietrich E. Wolf. "Cellular automata approach to three-phase traffic theory." Journal of Physics A: Mathematical and General 35.47 (2002): 9971.