Social and Technological Networks Report

Transportation System for Self-driving cars and buses

POUGALA Biko - s1651792

November 2019

1 Abstract

This project centers on shared on-demand autonomous vehicles in metropolitan areas. We'll particularly emphasise the constraints and obstacles such systems must face in order to be practical and safe to use for users, as well as become a serious alternative to the individual car ownership model that's still prevalent in most large metropolitan areas around the world.

2 Introduction/Motivation of work

There are reasons to believe that an economic model based on on-demand autonomous vehicles will be prevailing in the near-future in most large metropolitan areas across industrialised nations, replacing the individual car system we have today. This would entail massive new challenges for our current transportation infrastructure, and that for several reasons we hereby detail:

- 68% of the world population will be living in cities by 2050 compared to 55% today, according to the UN's Department of Economic and Social Affairs. In order to absorb that increase in transportation demand, it would entail considerably expanding urban areas' road networks to accommodate for the millions more cars that would be needed.
- American think-tank ReThinkX said this about the current trends in autonomous vehicles: *"We are on the cusp of the fastest, deepest, most consequential disruption of transportation in history. By 2030, within 10 years of regulatory approval of Autonomous Vehicles (AVs), 95 percent of U.S. passenger miles traveled will be served by on-demand autonomous electric vehicles owned by fleets, not individuals, ..."*

3 Related work

Talk about what traditional car makers and disrupters are doing and how much they are investing to get to the point we want to achieve.

4 Model and Problem statement

We use the San Francisco roads network developed by the University of Utah using the theoretical approach from Dr. Thomas Brinkhoff. This came under the form of unpractical `.cedge` and `.cnodes` files which I copied line by line into a `.txt` file before running a Python short script to convert it into a CSV file.

I created two CSV files, one named `sf_edges_network.csv`, listing all the edges in the SF road network graph, with each edge endpoint corresponding to a node ID as well as the L2 distance between the two nodes and a second file called `sf_nodes_network.csv`, listing all the nodes in the SF road network graph, where each node ID corresponding to a pair of latitude and longitude coordinates. For reference here is how they look: I have converted this DataFrame

	edge_id	start_node_id	end_node_id	l2_dist
0	0	0	5	40.444225
1	1	0	2	10.284244
2	2	0	1	5.394655
3	3	118	121	10.404842
4	4	118	119	7.169434

(a) Graph representation of edges

	node_id	x_coord	y_coord
0	0	1905.934692	2760.598145
1	1	1910.898193	2758.484863
2	2	1915.310181	2764.824951
3	3	1915.861694	2765.529297
4	4	1918.619141	2754.258301

(b) Graph representation of nodes

Figure 1: Graph representation of the San Francisco road network

into a an actual graph by creating a Python `Graph` object. This is a massive graph with over 174,000 edges and around double the number of vertices. Our case scenario is a group A of people, spread around the city and a group B of autonomous cars, also spread, presumably randomly across the city. For the sake of the argument, we'll consider that `size(A) >> size(B)`. Everyone in A tries to order a car at the same time. Due to the limited number of cars available, they would have to share cars. We wish to explore, in this project, different algorithms to assign cars to as many users simultaneously at the same time, as well as assess their possible real-life performance and drawbacks.

5 My approach

Because this graph is massive, very sparse and loosely connected, it was really tricky at first to initialise random positions to cars and users in the graph as almost every time there would be no path between the two nodes. This is likely due to the very high number of dead ends and restricted areas as well as the fact that the algorithm doesn't allow U-turns. I therefore decided to contnuously draw random numbers between 1 and 172776 (the maximum node) and see if the two have a path. I ran the function 1 million times to get a suitable number of paths for the experiment. The code is implemented in the code cell named

"Initialiser for user positions" in the second Jupyter notebook. Here are the steps of my algorithm:

- Draw n numbers uniformly at random from the list of acceptable nodes to designate the starting node IDs of users. Their corresponding end nodes would be their destination.
- Draw m numbers uniformly at random from any node of the graph to designate the starting nodes of cars.
- Assign ear car to their closest user. For that, for each car get their x and y coordinates from the `sf_network` DataFrame. And compute the L2 distance with the x and y coordinates of each user then take the minimum. Once a user has been assigned to a car, remove it from consideration.
- For each car, run the Best-First Search algorithm to find the list of nodes it needs to go through to reach the destination node.
- Because $m < n$, at this point a considerable number of users haven't been assigned any car. Each step of the algorithm from now on corresponds to the move from one node of the graph to the other, as it tries to reach the destination node of the user it's been assigned to.
- At each step, for each car, repeat point number 3 for the remaining passengers and run the heuristic again to assign previously unassigned users to their closest car. This operation stops when all users have been assigned to a car or when all cars reach their destination, whichever comes first.

6 Results

My algorithm, implemented as `runAlgorithm()` in the second Jupyter notebook is a very preliminary and unpolished version of what a good path-finding algorithm should look like. The algorithms work surprisingly fast given the large input data: the entire operation can be run in less than 20 seconds when using 10 cars and 300 users as input. But there are several issues with my approach: users and cars have their location randomised at each new start of the process, which is not a realistic scenario, as urban areas are organised structures, some places are much more likely to attract potential users than others (depending on time, weather and occasional large gatherings like sporting events, conferences...). How would the algorithm perform when a significant number of users all demand a car in a very small area? The second point is that this algorithm picks up users as it goes and drops them off one after the other in order, but that could generate significant detours and delays for some users. The third point is that the path-finding algorithm is very limited in applications. More data should be included to decide which path to choose including live traffic information, weather and possibly a machine learning model that would learn which routes are less likely to be jammed given the time of the day.

7 Appendix