Computer Science 4ZP6 - Final Report

Machine Learning Vehicle Simulator

Team Members:

Hassaan Malik

Navleen Singh

Paul Warnick

Trevor Rae

Supervisor:

Dr. George Karakostas

Instructor:

Dr. Christopher Anand

About the Team

Hassaan Malik - Lead Designer

I am the type of person that is a very quick learner. I pick things up very quickly because of my strong technical background knowledge. I always strive towards finding the most efficient way of doing something and making sure I achieve it.

Trevor Rae - Back-End Architect

In the world of software development, I'm known as the Eighth Wonder of the World. I am not a quitter and stay on that one task until it is complete.

Navleen Singh - Lead Developer

A machine learning and algorithm developing expert, I'm the type of person who will always finish a project on time and above expectations. I strongly believe in learning by doing, and taking education into my own hands.

Paul Warnick - Head of Documentation

I'm a software engineer and business enthusiast. In my spare time I enjoy working on useful personal projects, learning web development skills and doing just about anything involving accounting and personal finance.

Introduction

With autonomous vehicles becoming more prevalent on today's roads the ability to optimize traffic flow by adjusting vehicle software is more in demand than ever. Our team is developing such an algorithm that will be distributed to all smart vehicles on a network. Once activated, the vehicle will automatically take in a number of different inputs such as current traffic conditions and provide the car's navigation system with the optimal route to reach its destination.

In order to reach maximum vehicle throughput of the network we've decided on the following areas of focus for the algorithm:

Primary congestion avoidance

- Using machine learning (see Smart Car Approach below) our algorithm will be able to understand which routes will lead to the greatest amount of congestion and react by instructing the car how to avoid these situations.

Secondary congestion avoidance

- Here we increases efficiency on multi lane roads without changing a vehicle's current route
- Our algorithm will do an analysis of all vehicles ahead of the current car (within a given range) and determine the optimal lane to reside in

Technical Structure

In order to properly test the efficiency of our algorithm, our teams is creating a web based simulation platform built using the MEAN stack (tutorials found below):

- https://www.youtube.com/watch?v=PFP0oXNNveg
- https://auth0.com/blog/real-world-angular-series-part-1/
- https://www.tilcode.com/deploying-a-mean-stack-app-to-heroku/
- https://devcenter.heroku.com/articles/mean-apps-restful-api

The technical structure of the project is as follows:

- A Node.js backend controller that handles the majority of the calculations including vehicle movement, collision prevention, car orientation, routing and the Smart Car algorithm calculations

- A client-server websocket framework (Socket.io) for sending car data (X-Position, Y-Position and Orientation) and map data (edges and nodes that make up the city) to the front end
- A JavaScript front end for handling data and displaying it to the user in a clean and intuitive manner

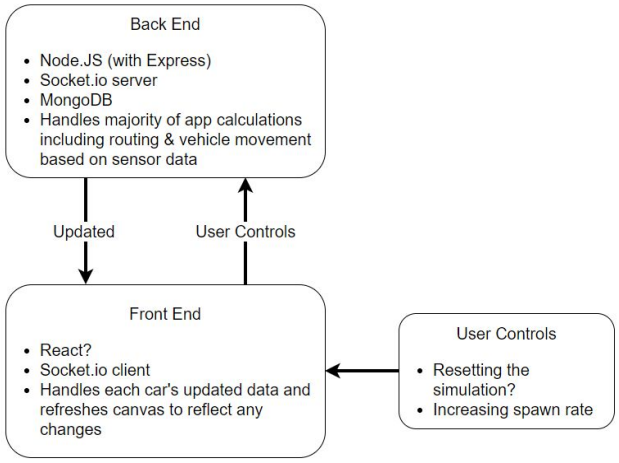


***Figure 1**: Technical structure of the simulator*

Dumb Car Approach

As a testing environment we've created a web application for simulating different variations of the algorithm. This application is essentially a model city built from a network of road objects allowing us to check against real-world conditions like heavy traffic and inner-city highways.

Within the simulation we've allowed for a percentage of our vehicles to be labeled as Dumb Cars. These DC's attempt to simulate regular human driving behaviour in the following two ways:

- For routing we've used Dijkstra's shortest path algorithm (similar to how a regular GPS would route a vehicle)
- Lane changes are only done if they are needed in order to make a left or right turn (ex. No left turns from the right lane)

Smart Car Approach

Our next major goal will be to implementing Smart versions of the aforementioned vehicles. We hypothesize a significant increase in car throughput via primary and secondary congestion avoidance. Our intent is to use a common machine learning technique known as Q-Learning in order to teach the vehicles how to behave in specific conditions. This technique uses what's known as "Reinforcement Learning".

An example of Reinforcement Learning is when a child learns to walk. The child will take a big step forward, then falls. The next time, it takes a smaller step and is able to hold its balance. The child tries variations like this many times; eventually, it learns the right size of steps to take and walks steadily. It has succeeded.

There are 3 basic concepts in reinforcement learning (state, action, and reward). In our situation the car will learn the best routes to get from point A to point B as fast as possible, it will also learn which lane is the best lane to travel in during certain situations.

Currently Reinforcement Learning is used for Playing the board game Go (finding the most optimal solution), Robot Control (finding the most optimal path, changing directions, etc), and many more.

The Q in Q-Learning stands for the long-term value of an action. Q-learning is about learning Q-values through observations. The procedure for Q-learning starts by initializing Q-values to 0 for every state-action pair: Q(s,a) = 0 for all states ***s*** and actions ***a***. Once the car starts learning, it takes an action ***a*** in state ***s*** and receives a reward ***r***. It also observes that the state has changed to a new state ***s'***. The car will then update its values for state and action. When updating the values Q carries memory from the past and takes into account all future steps. We will use the maximized Q-value for the new state, ultimately resulting in the vehicle taking the optimal path.

Below we can see two different approaches to the Q-Learning algorithm:

```
if Terminal(s) then Q[s, None] ← r'
    if s is not null then
        increment Nsa[s, a]
        Q[s, a] ← Q[s, a] + α(Nsa[s, a])(r + γ maxa' Q[s', a'] - Q[s, a])
        s, a, r ← s', argmaxa' f(Q[s', a'], Nsa[s', a']), r'
return a
```

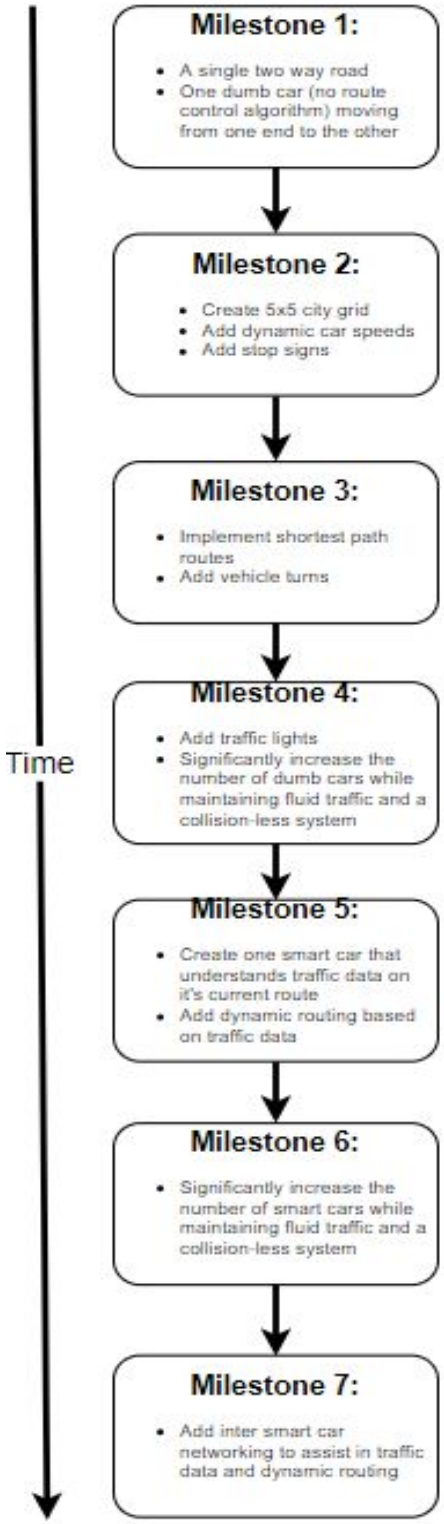
***Figure 2**: An example Q-Learning algorithm for analyzing a network*
***Inputs**: The current state of s' and reward signal r'*
***Persistent Values**:*

Q, a table of action values indexed by state and action, initially zero
Nsa, a table of frequencies for state-action pairs, initially zero
s, a, r, the previous state, action, and reward, initially null

```
Initialize Q($s, $a) arbitrarily
Repeat (for each movement):
    Initialize $s
    Repeat (for each step of movement):
        Choose $a from $s using policy derived from Q()
            (e.g., $e-greedy)
        Take action $a, observe $r, $s'
        Q($s, $a) <-- Q($s, $a) + $Array[$r + $y * max($Array), Q($s', $a') - Q($s, $a)]
        $s <-- $s';
```

***Figure 3**: Another Q-Learning algorithm similar to above*

Implementation Plan



Expected Results

Given our current stage of development, testing the simulation with only Dumb Cars has resulted in a number of different insights. When reaching vehicle saturations above 40% (approximately 20+ vehicles per road segment) we notice a steep increase in average trip duration. Additionally, the vehicles show no signs of attempting to disperse over the road evenly, instead just focusing on the lane they should be to complete the next turn.

With these in mind, we expect the deployment of smart vehicles to increase overall throughput percentage by the following formula:

$$f(x) = C * \frac{d}{t} + \frac{s}{t} * x$$

***Figure 4**: Formula for throughput percentage increase*

***Where**:*

f(x): The current throughput of the network as a percentage of its maximum throughput

x: The efficiency factor of smart cars (0 being as efficient as dumb cars, 1 being perfectly efficient)

C: A constant value based off dumb car congestion levels (40% in our simulation)

d: Total number of dumb cars

s: Total number of smart cars

t: total number of cars

***Figure 5**: An example of sensor range for Smart Cars*

Conclusion & Future Work

Our project effectively demonstrates advanced machine learning algorithms that direct cars with increased precision. We simulate this increase in performance across a widely scalable application. Our project simulates independent learning as well as collective hive learning with varying results. In the future it is our hope that our simulation can be expanded to include a variety of additional algorithms and machine learning models with the hope of eventually identifying and creating the most efficient traffic modeling software possible.

Acknowledgements

We would like to thank Dr.Karakosta for this support and guidance throughout the duration of our project.

References

- Hu, Junling. “Reinforcement Learning Explained.” O'Reilly Media, 8 Dec. 2016, www.oreilly.com/ideas/reinforcement-learning-explained.
- “Autopilot.” Autopilot | Tesla Canada, www.tesla.com/en_CA/autopilot.