Investigating Algorithms to Manage Traffic Signals

James Frucht, Isaiah Turner

Abstract
Traffic is becoming an increasingly major problem that results in increased stress on commuters and increased emissions of pollutants. One form of mitigating these effects is using modern developments in computing and artificial intelligence to make traffic signals more efficient to limit the amount of time driver spend waiting at traffic signals. We developed a series of algorithms based on current implementations and new artificial intelligence methods, which we tested using the SUMO open-source traffic simulation software. We noticed significant improvements compared to current methods of managing traffic signals. We were unable to finish all of the algorithms that we initially intended to create due to time constraints.

Contents

1. Introduction

In the United States, traffic leads to many ecological, sociological and economic problems. Traffic congestion has reached all time highs, causing commuters to waste time and gas and be stressed. According to the Texas A&M Transportation Institute, US commuters spend an average of 42 hours per year delayed by traffic traveling for 22% more time than they would in conditions with no traffic. This wasted time driving leads to 19 gallons of wasted fuel per commuter per year and a cost of $960 per commuter per year. This amounts nationally to 6.9 billion hours spent delayed in traffic per year, 3.1 billion gallons of wasted fuel per year, and a cost of $160 billion per year[1]. The problem is only worse in the local area: Washington, DC, where, according to the 2018 INRIX Global Traffic Scorecard, commuters spend 61 hours per year in traffic congestion, costing each individual driver $1,694 per year, and the city of Washington $2,963 million per year[2]. These are significant time and economic costs on people, especially in the Washington, DC metro area.

Additionally, traffic can deteriorate mental and physical health. Kahneman et al. discovered in a study of nearly 1,000 women that commuting is their least enjoyable part of their day[3]. Another study by Anderson et al. found that people would rather give up five minutes of a leisure activity than spend a minute in traffic[4]. In terms of physical health, Hoehner found, longer commutes are negatively associated with physical activity and positively associated with BMI, waist circumference, systolic and diastolic blood pressure, and continuous metabolic score[5]. By reducing traffic congestion, the mental and physical health of many could have many could be improved by limiting the time and stress of driving, a sedentary behavior.

Since the invention of the automobile in the early 20th century, people have been looking for ways to make intersections more efficient: from the initial manned booths, to timers, and road-sensors. While these advancements transformed how we change lights, they have not affected why we change lights. For the most part, stop lights are changed with the goal of moving as many cars as possible through a specific intersection or small series of intersections. However, they fail to account for the fact that intersections are interdependent and the ultimate goal of an efficient traffic system is to maximize total movement throughout the system. With the advent of computing and specifically the recent increases in computing power and decreases in prices of computers as well as the necessary sensors for implementation, it is now possible to implement new solutions to improve stop light timings by accounting for interdependency.

A potential solution form of mitigating this problem is making traffic signals more efficient. By reducing the amount of time drivers spend waiting at traffic signals, commute times can be reduced significantly. We can model making traffic signals more efficient by minimizing the average commute time per driver (considering commutes remain constant), which is analogous to minimizing the following objective function:

$$f = \sum_{i=1}^{n} t_i$$

where n is the number of drivers t_i is the total time spent

traveling by automobile i.

In a real time environment, the total time spent traveling is unknown until the driver reaches their destination. The function above cannot be used to evaluate performance of a self-learning algorithm. As a result, we define another function to be maximized:

$$f_2 = \sum_{i=1}^{n} v_i$$

where v_i is the average speed of automobile i.

2. Methods

Our research consisted of developing a series of algorithms using both current methods and potential new algorithms with the goal of minimizing the two objective functions.

We used the Simulation of Urban Mobility (SUMO) software. SUMO is an open-source software developed by the German Aerospace Institute that models vehicle and pedestrian traffic. The simulation provides a granular system that models individual vehicles, lane changes and traffic flow. It had the tools necessary to conduct the simulations for our traffic signal: create both artificial test-case road networks and import real-world road map data, manually control traffic signals through their API, and collect data on traffic efficiency.

Our algorithms consisted of both control algorithms and experimental algorithms. The control algorithms consisted of algorithms that are currently implemented in the real world. These required very little computation and involved no communication between traffic signals. Our two control algorithms were static timings and simple dynamic timings. The experimental algorithms, on the other hand, have not been implemented in the real-world. They were compute intensive and involved the communication of traffic data between signals. Our two experimental algorithms were dynamic timings with communication and reinforcement learning.

Our algorithms were:

2.1 Static Timings

This served as the first control case. Mimicking most traffic signals without induction loops, each light phase is given a specific amount of time for which it is active regardless of traffic flow. The algorithm calculates the time it would take a single car to travel each controlled stretch of road using the length of the road and the speed limit. These values are then scaled up to a 90-second traffic cycle length, which is common for many traffic lights. The scaling is completed fractionally. All the unscaled phase lengths are added, and each phase is represented as a percentage of this total sum. These percentages are multiplied by the 90-second cycle length. A phase that was 10% of the unscaled cycle would become 9 seconds after scaling. As a result, longer stretches of road are allocated more time to account for traffic buildup.

2.2 Simple Dynamic Timings

This served as the second control case. The algorithm was designed to mimic current sensing technologies implemented in traffic signals. Each lane has simulated induction loops placed beneath them. These induction loops are able to detects the number of cars in the lane at any given time. If a traffic signal detects that there are more vehicles in the red light lanes than the green lanes, it switches to the next phase. We also used maximum and minimum times for each phase to limit extreme cases where a phase may never switch. The algorithm is designed to prevent situations where green lanes are empty or sparsely populated with cars while red lanes are full.

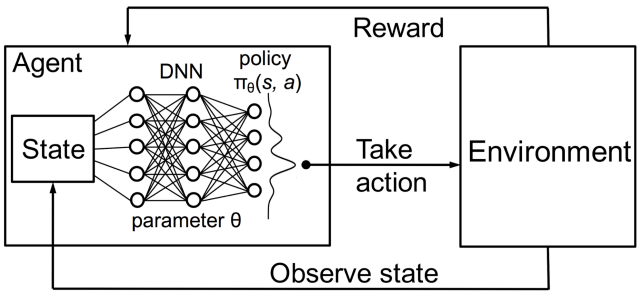
2.3 Dynamic Timings with Communication

Our third algorithm built off our second. The core functionality of the algorithm is the same. Lights still switch phases by comparing the number of cars in the red lanes to the cars in the green lanes. However, in networks with multiple lights, the traffic information from each light is sent to neighboring lights. The lights receive the information on potential incoming traffic and weight them when determining whether to switch phases or not. The numeric value of incoming cars is multiplied by a weight parameter to simulate the distance of incoming cars compared to the traffic waiting at the light. The weighting for the cars at the light was 10, and the weighting for the incoming cars was 1.

In addition, we wanted to maximize improvements in algorithm 3. Unlike the simple road length timings, we had adjustable parameters for algorithm 3, the maximum and minimum phase length time. In order to find the optimal values for these parameters, we developed a simple hill climber program. The program was designed to run algorithm 3 on a given network, then adjust the parameters and repeat the run. If changing the parameters resulted in an improvement in speed, that run was used as a baseline for further iteration. The program stopped when altering the parameters only slowed down the run. As a result, the hill climber found optimal parameters for algorithm 3 for each of our networks.

2.4 Reinforcement Learning

Reinforcement learning (RL) is a form of machine learning where agents take actions in an environment (see figure below). Reinforcement learning can be applied to many fields such as economics and game theory. Many recent developments in artificial intelligence have involved reinforcement learning, such as Deepmind's AI's that defeated professional players in the highly complex strategy games, Go and Starcraft. Due to these successes, we believed a similar AI could be used to solve the problem of optimizing traffic signals.



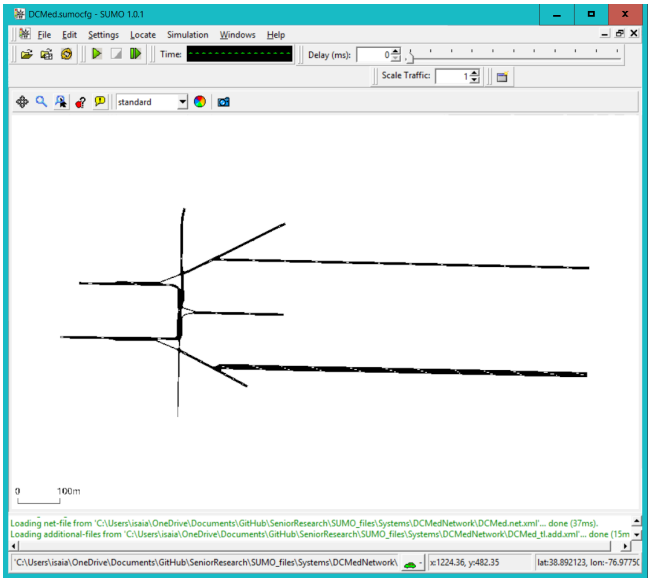
Reinforcement learning consists of the interactions between an environment and an agent. The environment depends on the specific problem. If someone were to work on a reinforcement learning algorithm for the game Pong, the environment would be the Pong video game. In our case, the environment was the SUMO simulation. The agent on the other hand, makes decisions in the environment. The general cycle for reinforcement learning consists of an agent observing the state of the environment. Then, the agent weighs the different elements from the environment in a neural network to determine which action the action will take next. Then the agent takes that action in the environment and receives a reward based on how the action affects the environment. Using this reward, the agent adjusts the weights of the neural network.

For our specific situation, with our environment being a traffic simulation, the state which served as the input for the agent, or observation state, consisted of both the number of cars at each intersection and the current phases of the traffic signals. The range of possible actions the agent could take in the environment, or action space, consisted of each possible phase for each traffic signal. Our reward was the sum of the speeds of all the cars in the SUMO simulation. Using OpenAI's open-source Gym toolkit, we implemented these elements of the environment into a Gym environment. Gym is an interface that facilitates interactions between environments and agents. For our case, it allowed us to implement general reinforcement learning algorithms for our specific case.

Initially, for our agent, we intended to implement a standard reinforcement learning algorithm such as DeepQ or PPO. However, we soon discovered that these algorithms would not work for our case because we had a multiple-dimension actions space. This is to say, we intended for our agent to take multiple actions at one time, switching various traffic signals at the same time, but these standard reinforcement learning algorithm only work when the agent takes only a single action at each step. Because of this limitation, we eventually decided to use the Branching Dueling Q-learning (BDQ) algorithm developed by researchers at Imperial college London [6], which can take multiple actions during a single step.

2.5 Testing

We used SUMO's road creation system to build a variety of road networks. Some simple test networks were constructed to represent the basic types of intersections. We constructed a T-Intersection and a Y-intersection. In each of these networks, the intersection was controlled by a single traffic light. In addition to these simple networks, we used OSM WebWizard to import real-world sections of road networks into the SUMO simulation. We imported a 2-way intersection in Washington D.C., a large network of around 30 intersections from D.C., and a complex 4-way intersection from Little River Turnpike. These networks had one traffic light for each intersection. We applied algorithms 1, 3, and 4 to the test networks and the real-world networks in order to collect our results. The DC Medium network is displayed below:



To run the simulation, we used SUMO's Traffic Control Interface (TRaCI). TRaCI provides the ability to retrieve information from the traffic simulation and manipulate it at every time step. Time steps are discrete units of time as measured by SUMO. The simulation progresses in these discrete units, keeping them accurate regardless of the hardware or the real-world time needed to complete the simulation.

For each of the networks, we injected cars into the simulation as soon as the simulation began. The vehicle count and routing was kept consistent for each testing network. We recorded the number of time steps needed to completely clear out all of the traffic.

3. Results and Discussion

We expected algorithm 3 to outperform the static road length timings. We expected that as the network grew more complex and larger in size, these speedups would decrease. Larger networks require more movement for cars to reach their destination, and more complex networks add additional possibilities for traffic jams. We calculated the percent speedup using:

$$\%speedup = \frac{steps_{algorithm1} - steps_{algorithm3}}{steps_{algorithm1}}$$

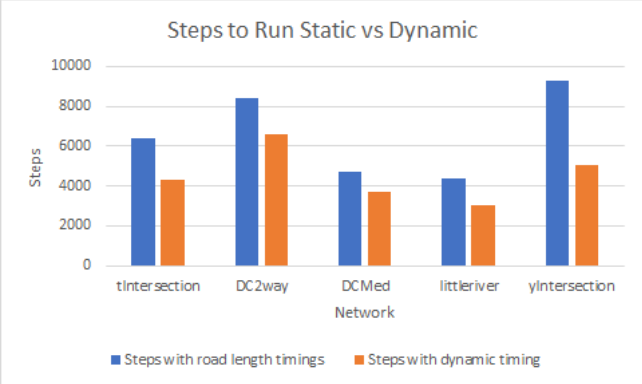


Figure 1. Traffic Speedup Graph

After completing our testing we compiled our results. The results for our comparison between algorithms 1 and 3 are shown below in table 1 and figure 2. The simple Y-Intersection and T-Intersection had the greatest speedups, of 36.33% and 32.37% respectively. Their simpler nature meant traffic was less bogged down by lane changes and turns into multiple available lanes. The DC 2-way intersection and the Little River Turnpike networks had smaller but still significant speedups. The large DC network had the smallest speedup out of all for the reasons stated above.

On the whole, the results for algorithm 3 were very promising. Every network had over a 20% improvement in time.

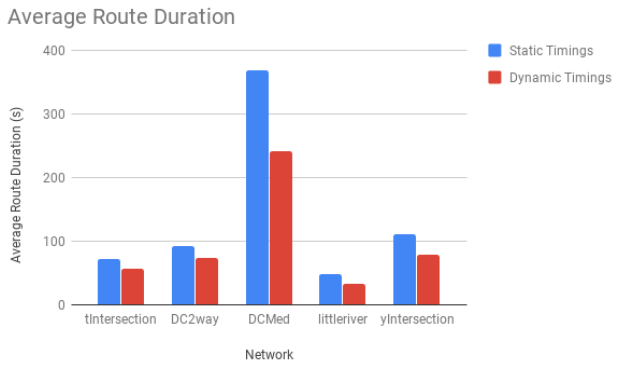


Figure 2. Average Duration Graph

We also tracked and graphed the average duration in seconds of the car's route. As the traffic pattern was the same regardless of the algorithm, these differences are due to improved traffic signals. Every network saw a reduction in average travel time. However, our DC network saw the greatest improvements. Unlike our single intersection networks, the DC network allowed cars to travel significant distances passing through multiple traffic lights. As a result, speedups were of a greater magnitude.

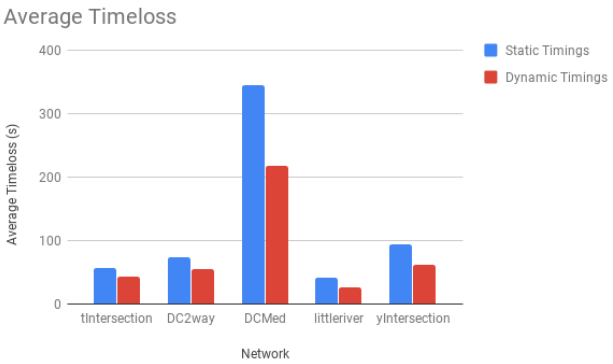


Figure 3. Average Timeloss Graph

Lastly, we measured the average timeloss across the network. Timeloss is a measure of the time lost to cars driving below their desired speed. In the SUMO simulation, the desired speed is the speed limit of the road. Timeloss provides an effective measure of how negatively cars are impacted by traffic. Once again, our algorithm resulted in decreased in timeloss. Cars wasted less time stuck in traffic. The difference was higher in magnitude in the DC network for similar reasons as the decrease in average route duration.

Although we were successful in implementing the environment and neural network, we did not have the time to fully train our reinforcement learning algorithm. As we trained though, we saw gradual improvements to the efficiency of the traffic. However, the network began with random weights. Even the improvements we saw when testing were not enough for any significant speedups. We would require more time to train and test our network.

4. Conclusions and Open Questions

Our investigation into algorithms for managing traffic signals showed that efficient traffic signals that use communication between signals can be an effective way to reduce traffic congestion, decreasing the amount of time drivers spend traveling. This has the potential to greatly reduce costs on drivers by limiting gasoline consumption and wasted time. Additionally, it can improve mental and physical health by reducing stress and sedentary behavior.

However, these various benefits would come with many costs that would have to be accounted for when deciding to implement these traffic light algorithms. First, these traffic signals would require significant infrastructure development for creating fast and stable network connections between traffic signals and installing vehicle sensors at all traffic lights. Additionally, these algorithms are far more compute-intensive than current methods of traffic signal management. As a result, traffic signal computers would need to be upgraded and, for centralized algorithms, resources would need to be allocated to running a secure server. The algorithm also requires induction loops or some other method of detecting the number and location of the vehicles at a traffic light. To properly assess

Network	Steps with algorithm 1	Steps with algorithm 3	% speedup	Network type
T-Intersection	6371	4309	32.37%	Small
DC 2-way	8380	6557	21.75%	Small
DC Medium	4692	3680	21.57%	Large
Little River	4378	3028	30.84%	Small
Y-Intersection	9302	5048	45.73%	Small

Table 1. Traffic Speedups

the effectiveness of implement more efficient traffic signals, more research must be done to determine the costs required to implement them, and whether they are less than the potential benefits.

A future project could complete the work on reinforcement learning. We were unable to train our network and gather results due to time constraints. If a group of researchers wished to use the same neural network architecture and Gym networks as we did, they could continue the training process.

Additionally, future studies could implement some of the functionality provided by SUMO to gather a more comprehensive view of the algorithm's success. By adding different vehicle types, public transportation, pedestrian movement, vehicle accidents, and other functions, researchers could simulate the algorithm's success in a more realistic scenario. SUMO also has a variety of methods to track vehicle emissions. A future study that defined vehicle types to create a realistic profile of traffic on our roads today could measure how emission levels change before and after our algorithms are applied.

5. Acknowledgments

We would like to thank all of the resources we were given in helping us with this project. We would like to thank Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker, the developers of SUMO, and OpenAI, the creators of Gym. We also would like to thank Ben Striner (bstriner on Github), who added a generic SUMO network-interface to Gym, which we used as a base for creating our Gym environment. Furthermore, we would like to thank Mihir Patel for guidance on machine learning and give a special thanks to Thomas Jefferson High School for Science and Technology and, specifically, the Computer Systems Research Lab, for providing us with the time and computing resources necessary to complete our research. We would also like to thank lab directors Mr. White and Dr. Zacharias for providing guidance and mentorship for our research.

References

[1] David Shrank, Bill Eisele, Tim Lomax, and Tim Bak. 2015 urban mobility scorecard. Technical report, Texas AM University Transportation Institute and INRIX, 2016.

[2] 2016 global traffic scorecard. Technical report, 2017.

[3] D Kahneman, AB Krueger, DA Schkade, N Schwarz, and AA Stone. A survey method for characterizing daily life experience: The day reconstruction method. *Science*, 306:1776–1780, 2004.

[4] Michael L. Anderson, Fangwen Lu, Yiran Zhang, Jun Yang, and Ping Qin. Superstitions, street traffic, and subjective well-being. *Journal of Public Economics*, 142:1–10, 2016.

[5] C.M Hoehner, C.E. Boarlow, P. Allen, and M. Schootman. Commuting distance, cardiorespiratory fitness, and metabolic risk. *American Journal of Preventive Medicine*, 42:571–578, 2012.

[6] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pages 4131–4138, 2018.