Traffic Signal Control: Deep Q-Learning with Various Rewards

Kyle Masters
USU Computer Science
CS5640

Abstract

Suboptimal traffic signals may cause traffic issues in congested networks and so it's important that they be optimized. This project experiments with different reward structures for a learning agent in a simple simulated intersection. Using a Deep Q-Learning agent, results are evaluated on the amount of vehicles moved through an intersection, the mean speed over a period of time, and the mean queue length over a period of time. With preliminary testing on these reward functions (hyper-parameters for the agent are not necessarily optimal), a reward function dealing with the wait time generated the model that was evaluated as best.

Introduction

This project looks at controlling the phases displayed at a traffic signal for a single simple intersection. This problem has applications for the real world due to impacts that traffic has on the real world. A suboptimal traffic signal may increase the travel time of vehicles in a traffic network which may also cause additional carbon emissions. Transportation contributed to 27% of greenhouse gases in the US and mitigating increases in this would be beneficial [

This problem is modeled in SUMO [2] (Simulation of Urban MObility) and implemented through the SUMO-RL environment [3]. Specifically, this environment models a single 2-way intersection with 2 lanes in each of four directions. Each step, the environment provides information on the amount of queued vehicles and total number of vehicles in each lane and takes input of 1 of 4 actions which represent the possible traffic signal phases. The question being asked in this paper is: How do different reward functions impact the results of a reinforcing learning agent? The code used in these experiments is hosted on GitHub [4].

Methods

To answer this question, a Deep Q-Learning agent was applied to the environment. This agent implements a policy neural network and a target neural network, a replay buffer used to store and sample batches of information from each step, and evaluates 4 different reward functions.

Neural Networks

These neural networks are simple dense neural networks each with 2 hidden layers of 256 nodes each. They are connected with ReLU activation functions between layers. Every 200 steps, the policy neural network is updated with 8 batches of 20 episodes. Every 800 steps, after the policy network is updated, the target network is updated to match the policy network.

Reward functions

The SUMO-RL environment has different reward functions implemented that can be used to train the agent. Some of these different reward functions were used to train the agent over several episodes each of several thousand steps. These rewards utilize some variables from the reward as follows:

- Queued vehicles – A vehicle is considered queued when it's speed is less than 0.1
- Wait time – Amount of time spent queued by a vehicle during a time step.

The reward functions applied are:

- Wait Time: Difference in total wait time from last step
- Speed: Average vehicle speed at each step
- Queue Length: Average number of queue vehicles in each lane at each step
- Vehicle Count: Average number of total vehicles in each lane at each step

As the simulation progressed, information was gathered at each step that was used to measure the effectiveness of each reward function. The final information that was used to measure the effectiveness is the total throughput over the course of an episode. This is measured as the total number of vehicles that exit the intersection after passing through the middle throughout a single episode.

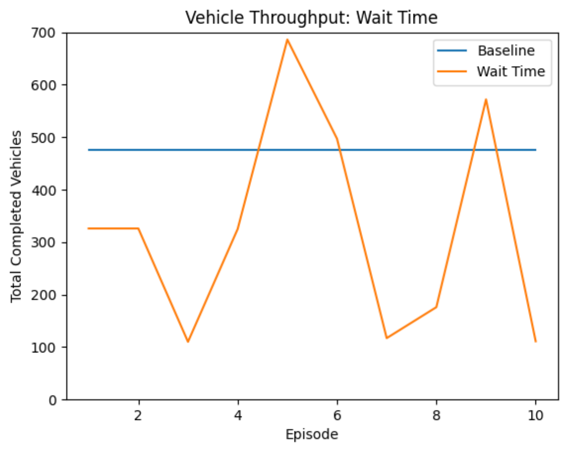
These experiments were performed over 10 episodes with each reward function. Each episode consisted of 5600 steps in which the policy network was updated 28 times and the target network was updated 7 times.

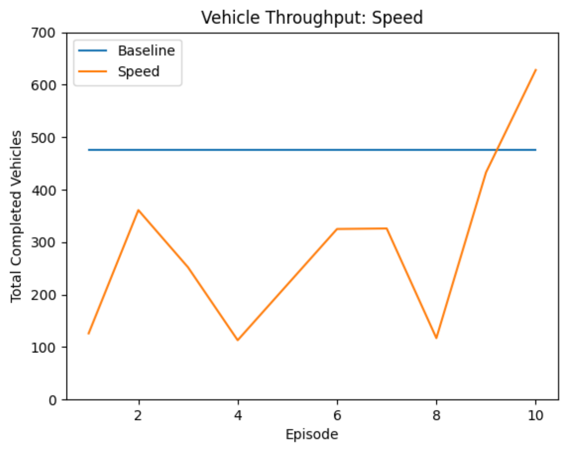
Results

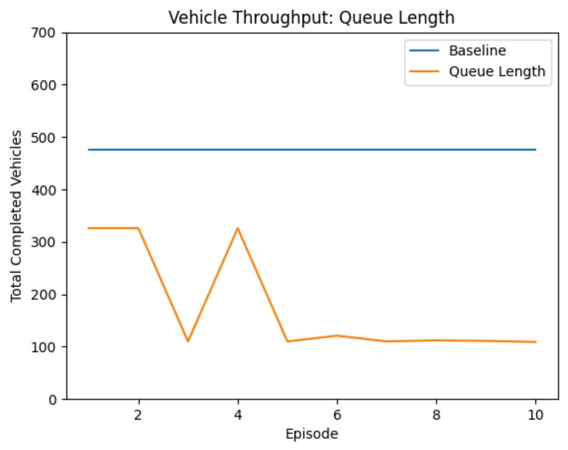
The results of the agent learning from these different reward functions were compared with a baseline traffic signal control method which cycled through the phases on a set timer. Both the maximum vehicle throughput from a model and the vehicle throughput in the last model from each reward are shown in the table below.

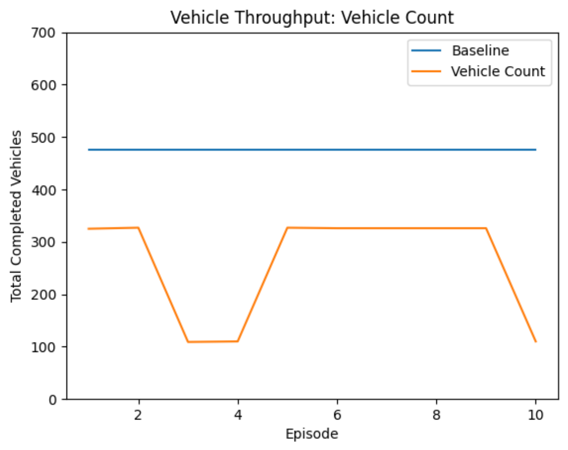
Method Name	Maximum Vehicle Throughput	Ending Vehicle Throughput
Baseline	476	476
Wait Time	**686**	111
Speed	628	**628**
Queue Length	326	109
Vehicle Count	326	110

Below the graphs plotting the maximum vehicle throughput are shown. In each graph, a different reward function was used and plotted against the baseline.









Summary

Conclusion

Compared with the baseline model, only the reward functions Wait Time and Speed performed well in generating a model that could be evaluated better than the baseline model. It was observed that among the models generated by the Wait Time reward function, there was a high amount of variance between results from model to model. The best performing model was generated at episode 5, with the model generated at the last episode performing significantly worse than the baseline model.

Some of these variations in the performance of the Speed models were noticed, but the general trend of those models was positive compared to the Wait Time models. The best performing model out of the Speed models was the last one, though further testing would be required to determine if this is conclusive that an agent using the Speed reward function learns more consistently than others.

The final two reward functions would be considered unsuccessful in these experiments. Both of them provided a maximum vehicle throughput of 326, about 68% of the vehicle throughput of the baseline model.

Overall, these models generated results that showed high variance from episode to episode. This may have multiple causes, including poor performance resulting from the architecture of the neural networks, poor choices in update periods for the policy and target networks, or general problems that could be attributed to using a DQN agent rather than another.

Future Work

These experiments could be significantly improved on in the future. The first area of improvement, with more computing power allotted, would be better hyperparameter tuning. Hyperparameters to tune would include the previously mentioned update periods for the policy and target networks, the layers, nodes, and activation functions within the neural networks, the learning rate of the optimizer, and the loss function used. Additional types of agents could also be utilized to see if there is a better type of agent that could assess these reward functions.

The other area that these could be improved upon would be expanding the scope of this experiment by extending the episodes and steps that each agent was trained over. Firstly, the models could be evaluated at smaller periods to generate results that might give a better picture on what is causing high variability between episodic results. The total amount of steps used for training could also be increased. Each of these models was trained over 56,000 steps. This amount could be increased with higher computing power, so that it could be computed in a reasonable time. This would allow the ability to agents to learn longer to check if high variation continued to occur or if learning stabilized.

Sources

[1] “Overview of Greenhouse Gases,” EPA, 16-May-2022. [Online]. Available: https://www.epa.gov/ghgemissions/overview-greenhouse-gases.

[2] M. Behrisch and R. Hilbrich, “Simulation of Urban MObility,” *Eclipse SUMO - simulation of Urban mobility*. [Online]. Available: https://www.eclipse.org/sumo/.

[3] L. N. Alegre, “Sumo-RL,” *GitHub*, 07-Dec-2022. [Online]. Available: https://github.com/LucasAlegre/sumo-rl.

[4] https://github.com/kyle-masters/sumo_rl_cs5640