Q-Table Based Traffic Light Control

Miguel Rust

Conducted as part of a project for USU CS 5640 Fall 2022

Abstract

This project is evaluating the effectiveness of Q-Table based reinforcement learning agents at controlling a simple intersection. The intersection used has a north lane, a south lane, a east lane and a west lane. These lanes meet at a simple four-way intersection. The agent has access to the number of vehicles queued up in each lane. Using this information the agent will learn a policy to control the intersection in an optimal way. Several different reward structures and agents are evaluated as part of this project. The reinforcement learning agents are benchmarked against a rule of thumb agent and a partially random agent.

Intro

Traffic light control has significant impacts on travel times, intersection throughput, idle times, and safety for both motorists and pedestrians. Therefore, traffic light control is an active area of research. A good policy will perform well in all of these metrics.

The problem statement for my project is the following:

Given the number of vehicles queued up in each lane and the time spent on the current time phase, find the choice of light phase such that it minimizes the time spent at the intersection, while minimizing crashes.

The state representation of the intersection is the following:

```
((Lane 1 vehicles queued: int,
Lane 2 vehicles queued: int,
Lane 3 vehicles queued: int,
Lane 4 vehicles queued: int
), Time steps spent on current phase: int)
```

Several different reward structures were experimented with and will be discussed later.

Methods

The reinforcement learning method employed by this project is a quality table. Several quality tables were experimented with.

- Light Phase Selecting Agent

The first quality table experimented with had the following structure `np.zeros(shape=(15, 6, 6, 6, 6, 3), dtype=np.float16)`. The first dimension corresponds to the time spent on the phase. The time spent on the phase is binned into slots that store an interval of 10 discrete time steps. The next four dimensions correspond to the vehicles queued up in each lane. If the number of vehicles exceeds 5, it's binned into the fifth slot. The final dimension corresponds to the action space. There are three light phases so there are three slots. The agent using this table performed quite poorly and struggled to learn.

- Light Phase Toggling Agent

The second quality table experimented with had the same structure other than having a different action space. It only has two actions in its action space. The first action is to keep the light on the current phase and the second is to switch the phase. I built an adapter to the environment to facilitate this agent. The adapter does not call the agent's act and update methods for 10 time steps, after each light phase change. During this time it selects the omitted action which is the yield phase. The 10 discrete steps gives the cars in the intersection ample time to get out of the intersection, before the agent is able to change the phase. Therefore, this agent only has to worry about the frequency at which to change the phase. This simplifies the problem the agent is solving. This agent performed much better.

The following baseline agents were also used. These agents do not use reinforcement learning and don't learn.

- Random agent

This agent chooses a random light phase every 20 time steps. It sticks with this choice until it chooses again.

- Sequential agent

This agent loops through the light phases in the following order north south green, yield phase, and east west green. It changes its phase every 40 time steps.

In addition, two reward structures were experimented with.

- Number of vehicles in intersection

This reward structure counts the number of vehicles moving the through the intersection during the time step and then subtracts off 50 per crash to compute the reward. The motivation behind this is that a policy with a high level of throughput will have more vehicles traveling through the intersection during a typical time step than a lower throughput one.

- Idle time

This reward structure sums up the idle time of all vehicles currently queued up and multiplies it by negative one it then subtracts off 50 per crash like the other policy. This reward structures is attempting to reduce idle time which will likely increase throughput.

Environment

I initially attempted to use the following Open AI gym https://github.com/beedrill/gym_trafficlight. I ran into some issues that I was unable to resolve. For this reason, I wrote my own simple simulator. My simulator is found at: . This simulator has the ability to support various different reward structure, logging formats and visualization options.

Results

The learning curve for the light phase toggling agent using the number of vehicles reward structure is below. The agent in this case was also restricted to only taking action every 10 time steps. The benchmark agents and a line of best fit is also included in the plots.

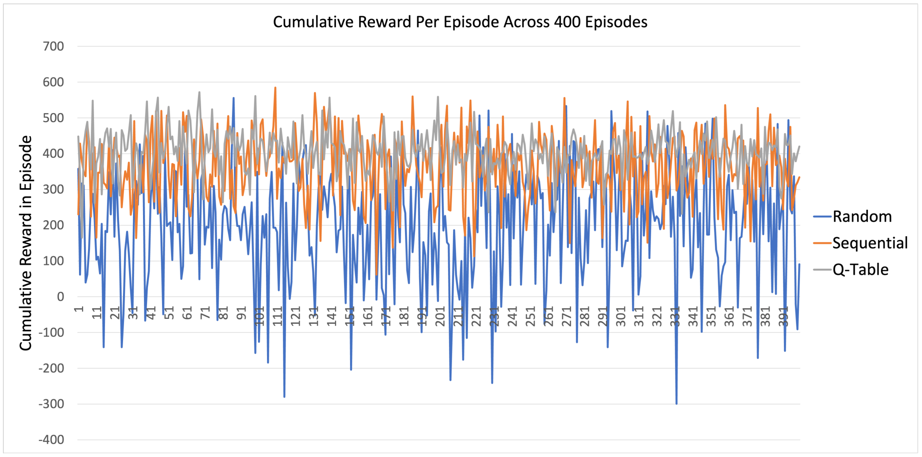

Figure 1: Figure 1

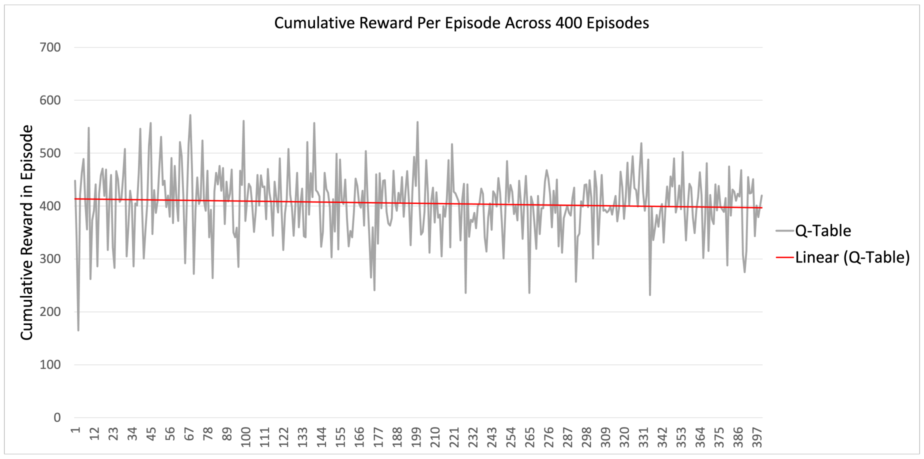

Figure 2: Figure 2

The agent performs well compared to the benchmarks, but is not learning. This difference is likely due to the restrictions placed on the action space for the reinforcement learner in this case.

The learning curve for the light phase toggling agent with the idle time reward structure is below.

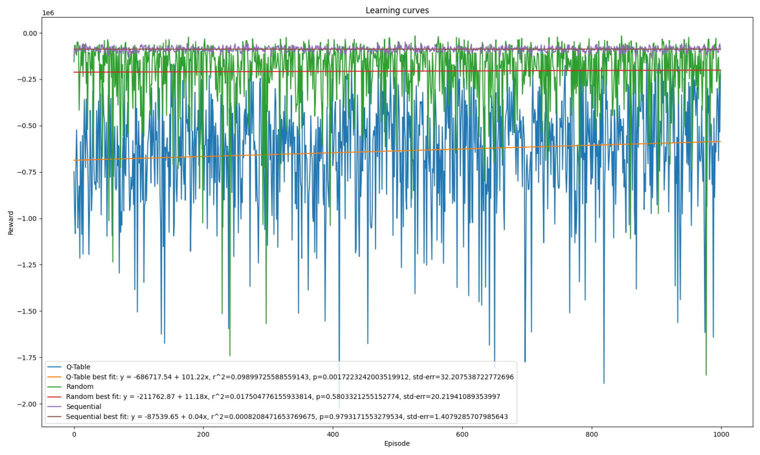

Figure 3: Figure 3

The agent in this case performs worse than the benchmarks, but is learning. The rate at which it learns is quite slow, but it's statistically significant. The slope coefficient for the line of best fit is statistically different than 0 at a confidence level of 0.01.

Summary and Conclusion

From my current results it appears that a Q-Table based agent is a poor choice for traffic light control. The rate at which it learns is quite slow and it performs poorly compared to simple rule of thumb agents.

Sources

- https://arxiv.org/pdf/2104.10455.pdf
- https://github.com/beedrill/gym_trafficlight