Q-learned traffic light systems

Johnny Wong

1 INTRODUCTION

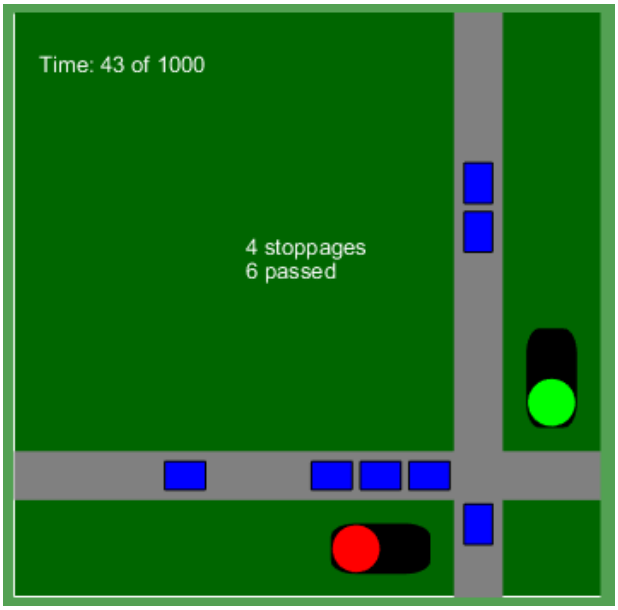
Traffic lights are a daily encounter for many people. They are crucial to the efficient coordination of transport, but their algorithms are not often thought about. The most basic algorithms simply cycle through various light settings on a fixed time schedule, while advanced algorithms are dynamic and change according to traffic conditions.

This assignment explores the use of reinforcement learning to teach traffic lights to respond to real time car positions. The different traffic light systems were then tested under many simulations

When the learned traffic light system was compared to the fixed time system, it was found to be far more effective in allowing traffic through, reducing the total amount of time vehicles were stopped at red lights.

2 IMPLEMENTATION

The experiment space consisted of a simple intersection of two one-way streets, one vertical and one horizontal. At any given time, the light will be green for one direction and red for another. A diagram of the environment is shown below.



Here, the cars on the vertical road are heading down the screen and face a green light at the intersection. The cars on the horizontal road face a red light so must queue until the lights turn green. 9 cars could fit on the road before the intersection. Traffic lights had to wait at least 3 time units before they could change from red to green and vice versa.

Each positioning of cars in the intersection is mapped to specific predefined states.

The algorithm used to optimise the traffic lights is called Q-learning. This generates a matrix that suggests the best action to take in any given state. The actions are either changing the traffic light or keeping it the same.

Definition of states

The states were defined by:

- Closest position of cars on vertical road (10 possible values)
- Closest position of cars on horizontal road (10 possible values)
- Traffic light setting (2 possible values, 1 for each direction that is green)
- Time since last light change (4 possible values from 0 to 3)

Thus, there are a total of 10 x 10 x 2 x 4 = 800 states. At each state, the system is penalised anytime a vehicle stops and gives a reward when a vehicle is passed through. When a car is the closest it can be to the intersection, but the light is red, a penalty of -1 is imposed. When the light is green and a car passes through the intersection, a reward of 0.1 is given.

$$R(s) = \begin{cases} -1 & if\ vehicle\ is\ stopped\ at\ intersection \\ 0.1 & if\ vehicle\ passes\ through\ intersection \\ 0 & if\ no\ vehicles\ are\ stopped\ or\ passed \end{cases}$$

Creation of Q-matrix

Creation of the Q-matrix was done under batch processing. Unlike online algorithms, the Q-matrix will not update once it has been created. During the creation phase, cars are randomly generated with a defined, independent probability of entering the system for each road. If the traffic light can change, then it will randomly choose whether to change or not. At each time step, the current state, action taken and next state are used to update the Q-matrix under the following formula:

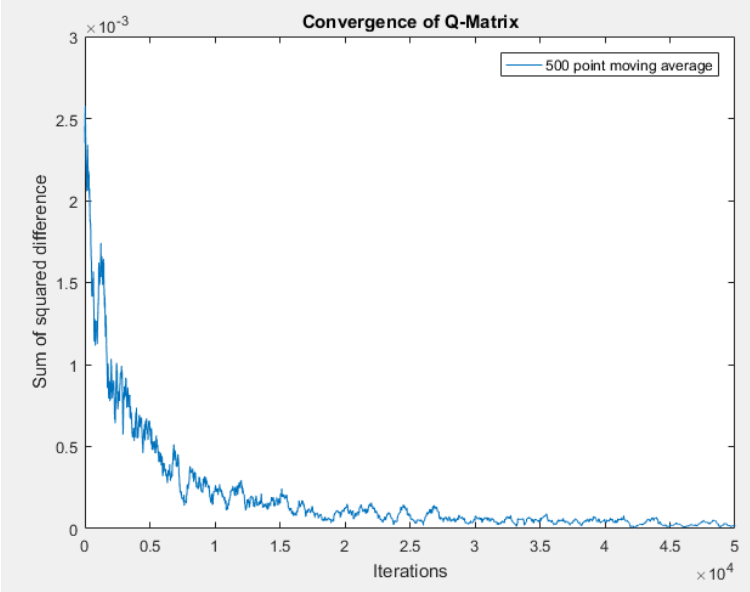
$$Q(s,s') = (1-\alpha)Q(s,s') + \alpha(R(s) + \gamma \max_{s''} Q(s',s''))$$

$s = current\ state$
$s' = next\ state$
$s'' = states\ that\ can\ be\ reached\ by\ s'$
$\alpha = learning\ rate$
$\gamma = discount\ rate$

This process was simulated 50,000 times to generate a matrix Q, whose elements Q(i,j) ultimately represent the future benefits of moving from state i to state j. To see whether the Q matrix was converging, the Sum of Squared Difference (SSD) was plotted to see the change in the elements between a Q-matrix, say Q_i and the Q-matrix after it was updated, Q_{i+1}.

$$SSD(i) = \sum_j \sum_k (Q_{i+1}(j,k) - Q_i(j,k))^2$$

The SSD is graphed on the right and shows that the Q-matrix does seem to converge after a couple thousand iterations.



Defining a decision rule

Once the Q-matrix is generated, the decision rule for the traffic lights is to take the action that leads to the state with the highest Q-element. Formally, if π(s) represents the decision rule at state s, and ζ(s,a) represents the state that the system will be in if it is currently in state s and takes action a, then

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s, \zeta(s, a))$$

Measuring results

The main metric used to compare performances of different traffic light systems was Total Stopped Car Time (TSCT) per cars passed through the system. TSCT is a measure of the total amount of time cars spend waiting and is calculated as:

$$TSCT = \sum_{i=1}^{T} Number\ of\ cars\ stopped\ at\ time\ i$$

Comments

The traditional algorithm for updating the Q-matrix includes a reward term that is dependent on both the current state and the action taken. In this experiment, the action did not have an immediate effect on the reward. This is because the scenario was programmed such that if a car is currently stopped at a red light, an action that would turn the light green would not allow the car to immediately advance. This mimics real life factors such as the reaction time of the driver. Similarly, if the car was already at a green light when it suddenly turned red, it would not be able to stop immediately. Hence, the action taken will not have an immediate impact, but a lagged effect on the respective Q-element.

Unlike other reinforcement learning problems, the way that these states have been set up doesn't allow for deterministic transitions. The states only account for the position of the closest car to the intersection and none that come after. Once the closest car passes the intersection, the system will enter some state dependent on the position of the previously second closest car, whose position cannot be known with the current state definition. Furthermore, the random generation of cars add further stochasticity to this process. Thus, the outcome of any action taken is not always foreseeable.

3 EXPERIMENTATION

3.1 PERFORMANCE OF SYSTEMS

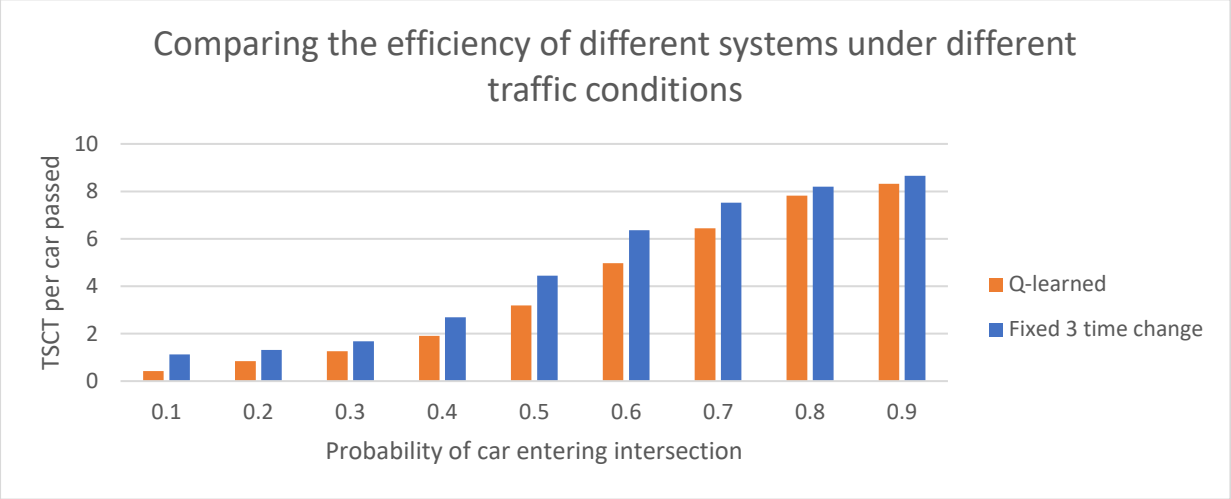
After generating the Q-matrix, it was put to test in a simulation. Starting with an empty intersection, 10,000 time points were simulated, with a 10% probability that a new car will enter the intersection from each direction at each time step.

The results are tabulated below.

	Total Stopped Car Time	Cars passed	TSCT per car passed
Q-Learned system	521	1938	**0.269**
3-time step change	2170	1955	**1.110**

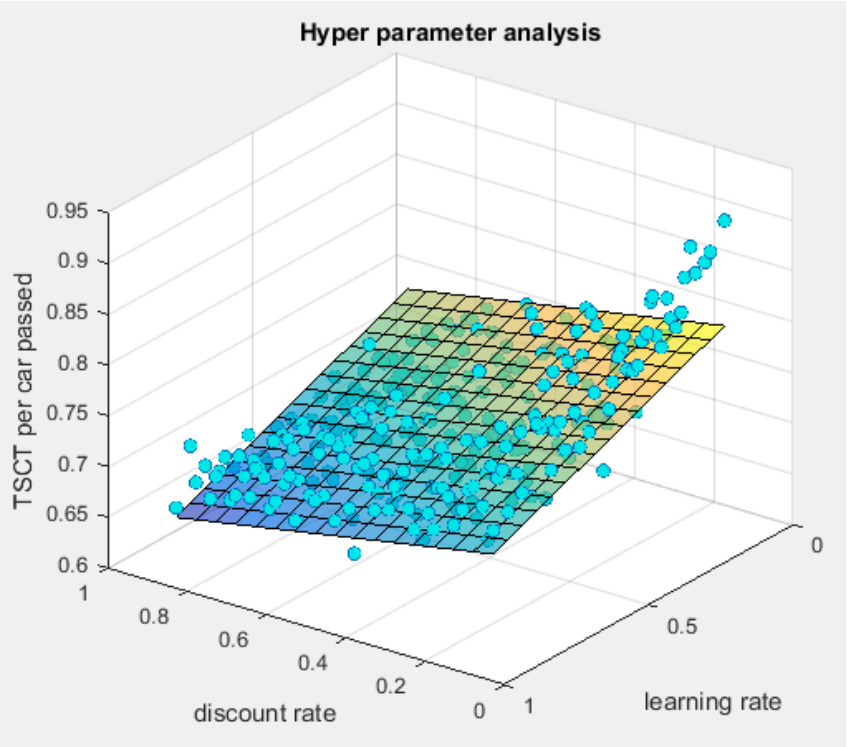
The Q-learned system performs much better, with a TSCT per car passed less than a quarter of that of the fixed time system.

This experiment was repeated with different traffic levels (different probabilities of cars entering the intersection) and is tabulated in the appendix (Table 1). In all cases, the Q-learned system performs better than that of the fixed time system. These results are graphed below.



3.2 HYPERPARAMETERS OF Q-LEARNING

Another factor of interest is the learning rate and discount rate. Simulations with different hyperparameter values were run and the resulting TSCT per car passed was modelled through a simple linear regression shown below (see appendix Figure 3 for coefficient estimates). Unsurprisingly, a higher learning rate leads to a quicker convergence. Interestingly, this did not come at the expense of performance, in fact, a high learning rate yielded statistically significant performance improvement. An explanation for this is that this model is not prone to overfitting, and a high learning rate would allow the Q-matrix to converge quicker, yielding better results. This is supported by Figure 1 in the appendix. Compared to Figure 2 in the appendix, the Q-matrix doesn't seem to completely converge with a small learning rate of 0.01.



A high discount rate yielded better performance, but did not have as much of an effect as the learning rate.

4 DISCUSSION

This project demonstrates the ineffectiveness of a fixed time light change traffic system. Specifically, it shows that a Q-learned system where states are defined by the proximity of the closest car in the horizontal and vertical direction performs much better. This section discusses some limitations to this state definition. It will then propose other potential state definitions that could be better or more efficient.

Limitation	**Proposal**
Only considers closest car - this state definition has no consideration of the number of cars behind the closest one	New state definitions that include the number of cars on each road
Large number of states - The current state definition has a high number of states - This causes the learning process to take longer - Computationally expensive	New state definition that only looks at which road contains the closest car instead of each car's specific position
Practical implications - The end goal of an analysis like this would be real life implementation - Implementing a system like this would presumably require the installation of sensors - Having sensors being able to detect each individual car's position along a road is not realistic	New state definition that counts the number of cars within a certain distance of the intersection without keeping track of their specific positions.
Changing traffic conditions - The Q-matrix is learned under specific assumptions of traffic density - Traffic conditions are subject to change - Using the wrong Q-matrix to control the traffic light can lead to results even worse than a fixed time system	Monitoring traffic conditions to determine the most likely traffic density. Choose pre-learned Q-matrix based on estimated traffic conditions. Also, an opportunity to implement an online learning system as opposed to the batch system.
Unrealistic intersection - One way traffic - Single lane - No turns - No amber light - No car acceleration	Create a more sophisticated simulation. Will need even more states, thus testing above proposals beforehand may help reduce number of states needed.

5 CONCLUSION

Traffic lights can learn to pass cars through an intersection better. This project demonstrates that a method of machine learning called Q-learning successfully improves the traffic light system compared to a fixed time light change rule. Although a very simple model has been used, it can be considered as a proof of concept for more sophisticated systems to be engineered. Further experimentations with different state definitions can improve the efficiency, realism, and applicability of the system, allowing Q-learning to be used on a larger scale.

6 REFERENCES

http://www.cse.unsw.edu.au/~cs9417ml/RL1/algorithms.html
http://mnemstudio.org/path-finding-q-learning-tutorial.htm
http://artint.info/html/ArtInt_265.html
https://www.youtube.com/watch?v=A5eihauRQvo
“Reinforcement Learning For Adaptive Traffic Signal Control”, Jeffrey Glick, 2015, Stanford University

7 APPENDIX

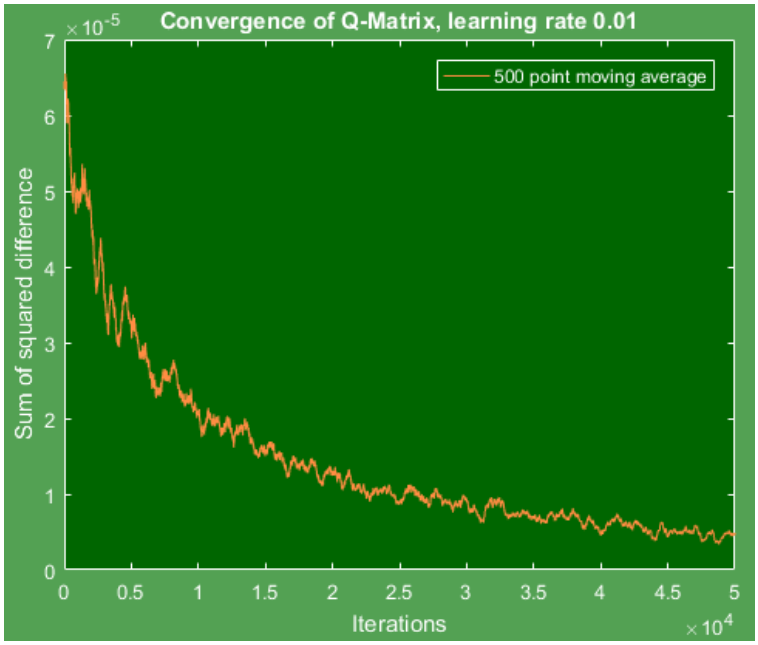


Figure 1: Convergence with 0.01 learning rate

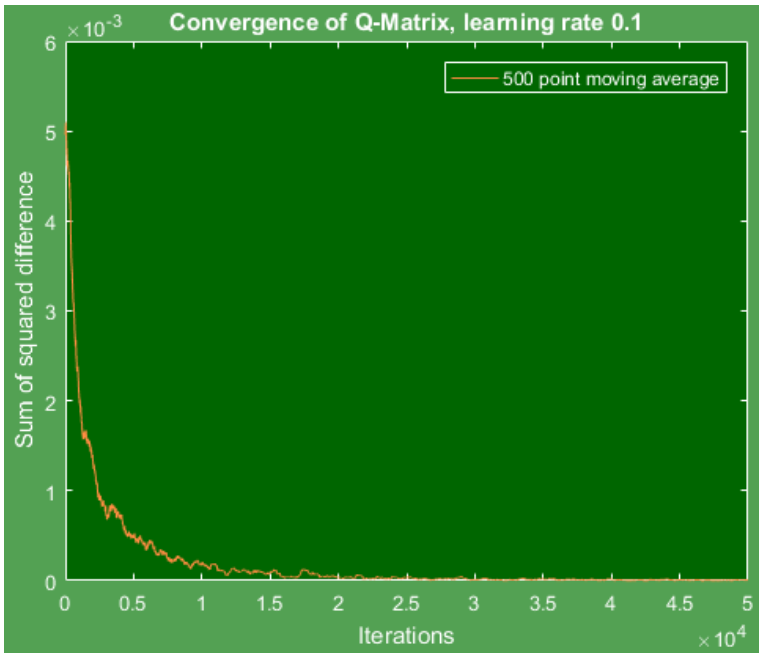


Figure 2: Convergence with 0.1 learning rate

	Coefficients	*Standard Error*	*P-value*
Intercept	0.819	0.00488	9.5143E-288
ALPHA	-0.123	0.00652	4.16041E-52
GAMMA	-0.070	0.00652	7.4616E-23

Figure 3: Multiple linear regression relating to hyperparameter analysis

Car Probability	TSCT per car passed Q-learned	TSCT per car passed Fixed 3 time change
0.1	0.4185	1.1215
0.2	0.8452	1.3147
0.3	1.2597	1.6854
0.4	1.9060	2.6855
0.5	3.1921	4.4413
0.6	4.9744	6.3561
0.7	6.4419	7.5241
0.8	7.8146	8.1946
0.9	8.3247	8.6614

Table 1: Comparison of systems under different traffic conditions