UM-SJTU Joint Institute

Decesion Making in Smart Cities
(ECE4530J)

Project Report
Q-Learning in Smart Traffic Light Control

Group 6

Ju Yiming	ID: 518370910059
Lin Yipeng	ID: 518370910060
Zhu Zhengping	ID: 518370910068

December 1, 2021

Abstract

This project targets at the optimal setting of traffic lights in an individual intersection, so as to minimize the congestion. Traditional traffic lights have fixed duration for red, amber, and green signals, which does not have adaptability to the random traffic. Reinforcement learning is considered as a promising strategy for adaptive traffic lights control. In this project, Q-Learning is used to learn and select the optimal actions for traffic light control in an isolated intersection that minimize the congestion. By comparing with the traditional traffic lights, we find that the Q-Learning algorithm do improve the congestion and are more adaptive to the randomness in the traffic.

1 Introduction

Nowadays, people are often caught in traffic jam, especially during the rush hour. The congestion is partly caused by the unreasonable settings of traffic lights. Most of traffic lights today work on simple timer, which means the colors change at a fixed time interval. This traffic light control policy is normally based on historical data, which cannot respond to real-time situations. This kind of fixed control sometimes may lead to heavy congestion in one direction even if its vertical direction is quite empty.

Our project aims to accomplish real-time control so that the time of traffic lights can be balanced automatically and then the conditions of congestion will be reduced. We simulate the intersections and control the traffic lights in terms of the real-time numbers of vehicles from different directions.

The rest of this report is organized as follows. Problem formulation and modeling is introduced in Section 2. Section 3 will explain the Q-Learning algorithm that we used, while the implementation and results will be described in Section 4. We will further discuss on the improvements that can be made in Section 6.

2 Problem Formulation

We model the road intersections and formulate the optimization problem. We assume that each intersection has two bi-directional roads with one lane each direction. For convenience, we define four directions to be {east=1, north=2, west=3, south=4}.

2.1 Data

The parameters we set in our problem:

- Coming probability of vehicles λ.
- Number of vehicles allowed to cross the intersection at each time slot n.
- Number of cars allowed in each lane K.

We let n equal to 1 and K equal 15 so that the number of vehicles crossing the intersection in each time slot is 1 and each lane has a maximum number of vehicles to be 15.

2.2 Constraints

Turning is not allowed in this problem and the number of vehicles in each lane is not allowed to exceed the capacity $K = 15$.

2.3 Variables

Assume that we only consider an isolated bidirectional intersection. Since each intersection has four directions, we define the number of vehicles waiting at the intersection $X_i(t), i = 1, 2, 3, 4$, where i denotes directions. Note that lane 1 are lane 3 opposite lanes, while lane 2 and lane 4 are opposite lanes. And $Y(t) \in \{0, 1, 2\}$ denotes the status of traffic light. The corresponding meanings are shown in the table, where $lights_1$ and $light_3$ indicate the east-west traffic lights, and $lights_2$ and $light_4$ indicate north-south traffic lights respectively. When $Y(t)$ equals to 0, the amber light will keep 2 time slots. The transition between other different light status is controlled by the algorithm, which will be introduced later.

Y(t)	$light_1$, $light_3$	$light_2$, $light_4$
0	amber	amber
1	green	red
2	red	green

Then with $i = 1, 2, 3, 4$, we define the number of vehicles appearing at the intersection to be $C_i(t)$ and the number of vehicles crossing the intersection to be $D_i(t)$. In terms of X_i, C_i, D_i, we can get the equations to update the variables in each time slot.

The queue size of next time slot equals to current queue size plus the number vehicles in and minus the number of vehicles out.

$$X_i(t+1) = X_i(t) + C_i(t) - D_i(t)$$

The number of vehicles crossing the intersection should be no more than 1 for each time step:

$$D_i(t) = \min(1, X_i), i = 1, 2, 3, 4$$

Specifically, if $Y_n(t) = 1$, $D_1(t) = \min(1, X_1)$, $D_3(t) = \min(1, X_3(t))$, $D_2(t) = 0$, $D_4(t) = 0$. If $Y_n(t) = 2$, $D_1(t) = 0$, $D_3(t) = 0$, $D_2(t) = \min(1, X_2)$, $D_4(t) = \min(1, X_4)$

2.4 Optimization Goal

Since the purpose of our project is to reduce congestion, we should minimize the average waiting time of each vehicle. The congestion cost in time slot t can be expressed as $F(X(t)) = \sum(X_i(t))^2, i = 1, 2, 3, 4$, where $X_i(t)$ is the number of vehicles in i direction. The total count of vehicles that appear in the intersection is $G(C(t)) = \sum(C_i(t)), i = 1, 2, 3, 4$ The objective function can be formulated as

$$\min \frac{\sum_{t=1}^{T} F(X(t))}{\sum_{t=1}^{T} G(C(t))}$$

3 Q-Learning Algorithm

3.1 Reinforcement Learning

In unknown environments, reinforcement learning is a machine learning method that can progress based on feedback from environment, namely rewards and punishments. It is a model-free, unsupervised method that enables the agent to learn an approximately optimal policy. The optimal policy, to be precise, is that how the agents take actions in the environment so that the overall reward gain is maximized. The reinforcement learning process is illustrated by Figure 1.

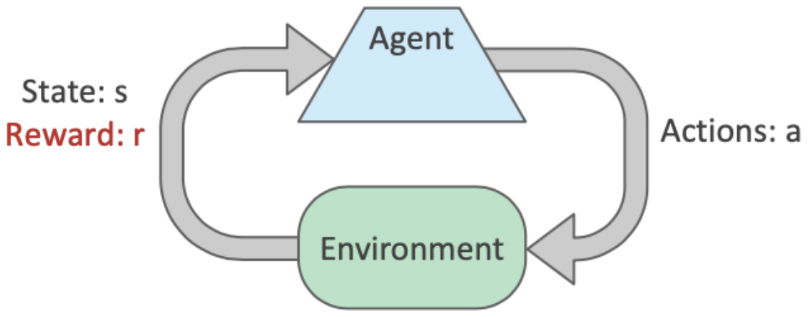


Figure 1: Reinforcement Learning Process

3.2 Q-Learning

Q-Learning is an online reinforcement learning algorithm that does not need the model of environment. The Q-Learning algorithm stores the reinforcement learning values and the corresponding state-action pairs in the Q-table. Optimal action that maximizes discounted cumulative rewards over time is then taken from the Q-table. We use ϵ-greedy method to choose actions from Q-table, namely the agent has the probability of ϵ to take a random action and $1 - \epsilon$ to pick the currently optimal action.

Denote the state where the agent is in at time step t is $s_t \in S$, the action it takes as $a_t \in A$, and earned reward as $r_t \in R$. Let γ $(0 \leqslant \gamma \leqslant 1)$ be the discount factor and α $(0 \leqslant \alpha < 1)$ be the learning rate. The Q-values (of state-action pairs) can be updated by the Bellman Equation:

$$Q^{new}(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot \max_{a \in A} Q(s_{t+1}, a))$$

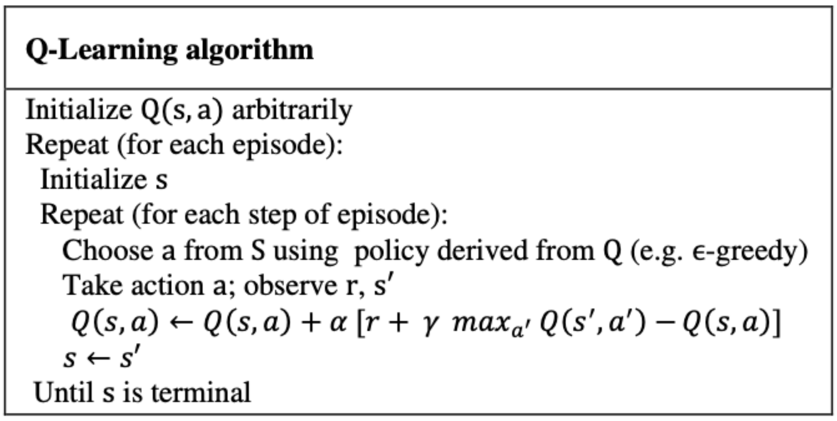
The pseudo code for Q-Learning algorithm is shown below:

4 Algorithm Implementation

4.1 Q-Learning States

In our project, we particularly focused on the optimal control of one isolated bidirectional intersection, namely we have index of intersection $n = 1$ and index of directions $i = 1, 2, 3, 4$. Therefore, the state s_t of the Q-Learning is defined as:

$$s_t = [X_1(t), X_2(t), X_3(t), X_4(t), Y(t), m]$$



Q-Learning algorithm

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from S using policy derived from Q (e.g. ϵ-greedy)
    Take action a; observe r, s'
     $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\ max_{a'}\,Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
 Until s is terminal
```

Figure 2: Q-Learning Algorithm Pseudo Code

Here $m = 0, 1, 2$ denotes the remaining time of the amber. To guarantee the time of training is acceptable, the maximum capacity of $X_i(t)$ is restricted to 15. A discussion will be made in Section 6 about this capacity.

4.2 Q-Learning Actions

Recall that the status of traffic lights is defined as follows:

Y(t)	$light_1$, $light_3$	$light_2$, $light_4$
0	amber	amber
1	green	red
2	red	green

The action of Q-Learning $a_t = 0, 1$ is then defined as:

Y(t)	a_t	Influence
1	0	$\mathrm{Y}(t+1) = 1$
1	1	$\mathrm{Y}(t+1) = 0$, $\mathrm{Y}(t+2) = 0$, $\mathrm{Y}(t+2) = 2$
2	0	$\mathrm{Y}(t+1) = 2$
2	1	$\mathrm{Y}(t+1) = 0$, $\mathrm{Y}(t+2) = 0$, $\mathrm{Y}(t+2) = 1$

Note that in a state that $\mathrm{Y}(t) = 0$, no action should be taken. In other words, the time of amber is fixed to two time steps in order to let vehicles in the intersection safely pass the intersection.

4.3 Reward Function

The reward (penalty) function we adopted is described as follow. The reward is affected by the action a_t, the light status $Y(t)$, the capacity of each lane K (here K is equal for each lane), as well as the number of vehicles X_i on each lane:

$$r_t = K + 0.3 \times P - Q^2/K$$

Here P and Q are depended on the action and light status. All situations are listed below:

a_t	$Y(t)$	P	Q
1	2	max($X_1(t)$,$X_3(t)$)	max($X_2(t)$,$X_4(t)$)
1	1	max($X_2(t)$,$X_4(t)$)	max($X_1(t)$,$X_3(t)$)
0	2	max($X_2(t)$,$X_4(t)$)	max($X_1(t)$,$X_3(t)$)
0	1	max($X_1(t)$,$X_3(t)$)	max($X_2(t)$,$X_4(t)$)

The term K is to ensure the reward function always have positive reward to encourage the system to move on. The term $0.3 \times P$ gives rewards on the lane that is going to be set to green light. 0.3 here is to reduce the reward comparing to the penalty to ensure the car numbers are less than capacity. The term $-Q^2/K$ gives penalty to the car number on the lane that is going to be turned to red light. The reason to have square here is to make the penalty less when there are less cars as we don't want the light to change that often due to the 2 time-step cost yellow light. And the division of K is to ensure the penalty is smaller than the capacity. The reward function should always be positive in this case.

Such reward function can have the following effects. It is encouraged to have longer green lights on lanes that have more vehicles. It will receive penalty if the light status changes when it is already green for lanes with more vehicles. The whole system is encouraged to have as few vehicles as possible while prevent number of cars from any lane to exceed the capacity.

4.4 Q-Learning Parameters

Taking the computer power into consideration, the following are the parameters for the Q-Learning:

α	0.5
ϵ	0.4
γ	0.5
Training Step	1,000,000

5 Results and Evaluations

After 1,000,000 of training, the model we get is stored in the Q-table. Evaluations are made by comparing our model with natural traffic lights when i) $C_i(t)$ is equal for $i = 1, 2, 3, 4$ (each lane has equal expectation of coming vehicles) and ii) $C_1(t) = C_3(t) \neq C_2(t) = C_4(t)$. The policy of natural lights is set as: change status every 6 time steps.

5.1 Equal Expectation of Coming Vehicles

For this part, $C_i(t)$ is set to have the expectation 0.3. Both the natural model and Q-Learning model are tested for 5000 times. For each time of test, 500 time steps are taken and the average cost is calculated. Recall that the average cost is determined by $\min\frac{\sum_{t=1}^{T} F(X(t))}{\sum_{t=1}^{T} G(C(t))}$. The box-and-whisker diagrams are plotted for the 5000 results of test for both Q-Learning model and natural model, as shown in Figure 3.

From Figure 3 we can find that, first, the Q-Learning model has less average congestion than the natural traffic light model. Second, the outliers of Q-Learning model have less deviation than those of natural traffic lights. This indicates that the Q-Learning model

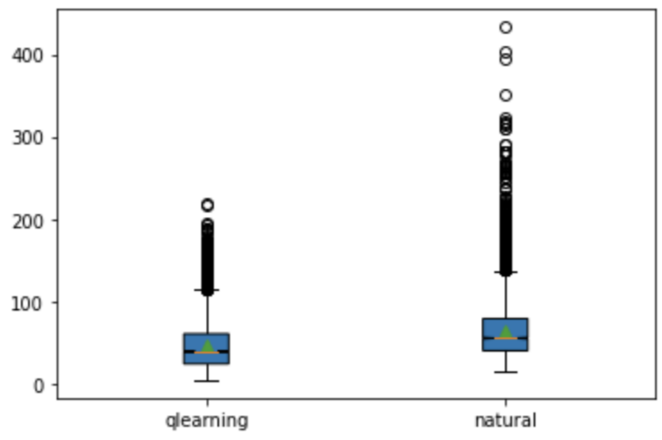


Figure 3: Box-and-Whisker Diagrams for Equal Expectation of Coming Vehicles

has the ability to adapt to some extreme situations (more vehicles come into a lane than into others) that happen randomly.

5.2 Different Expectations of Coming Vehicles

By assigning different expectations for different lanes, the adaptability of Q-Learning model can be better demonstrated. Here we set the expectation of $C_1(t)$ and $C_3(t)$ as 0.3, while the expectation of $C_2(t)$ and $C_4(t)$ are set to 0.1. After 5000 times of test, each of which has 500 time steps, we also plotted the box-and-whisker diagrams, shown in Figure 4.

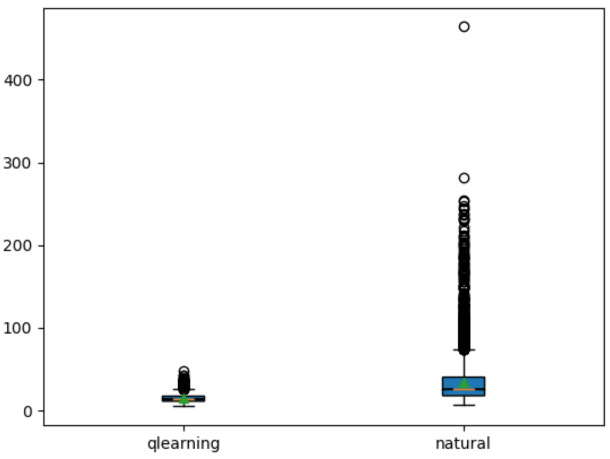


Figure 4: Box-and-Whisker Diagrams for Different Expectations of Coming Vehicles

From Figure 4 we can find that, the difference of average congestion between Q-Learning model and natural model is even greater. More importantly, as the different expectations of coming cars promise that more vehicles will come to some lanes than to others, the Q-Learning model shows great advantage in adapting to this situation.

However, the natural traffic lights lead to more outliers in the diagram and greater average congestion.

6 Conclusions and Discussions

6.1 Conclusions

In conclusion, by applying Q-Learning algorithm to an isolated bidirectional intersection control, we come up with a smart traffic light that is able to reduce the average congestion and to adapt to randomized coming vehicles in the intersection.

6.2 Discussions

Restricted by the computing power, the maximal capacity of each lane is set to be 15, and the size of Q-table is fixed based on the capacity. The problem is that, if the vehicles waiting in a lane have exceeded this capacity, we are unable to iterate on all states in the Q-table. We can take to options when encounter this situation. Either we may stop vehicles entering if the capacity has been reached, or we just let the program crashes. Both two options are just compromising, instead of solving the problem. Therefore, given that we don't have infinite computing resources, the Q-Learning model we made is only promised to work on situations when $C_i(t)$ is less or equal than 0.5, so that the capacity would not be exceeded. One possible solution to improve this is to use Deep Q Networks that makes approximations on the Q-Values, instead of making predictions based on the Q-table.

7 Appendices

1. S. Araghi, A. Khosravi, M. Johnstone and D. Creighton, "Q-learning method for controlling traffic signal phase time in a single intersection," 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2013, pp. 1261-1265, doi: 10.1109/ITSC.2013.6728404.

2. Palos, Peter & Huszák, Árpád. (2020). Comparison of Q-Learning based Traffic Light Control Methods and Objective Functions. 1-6. 10.23919/SoftCOM50211.2020.9238290.