ECEN689 Fall Dependable Learning System

Final Project Report

Traffic Light Control based on Q-Learning

Yi-Dan Jiang, Chun-Hao Huang

1. INTRODUCTION

Conventionally, the traffic light is control by pre-defined phase time. However, the conventional fixed-time control method is inefficient on condition that traffic flow is unbalanced between different directions. In the project, we use multi-agent systems and Q-Learning for traffic light control where the intersection saturation is used to define different states. Also, we design our actions and reward function to allow the traffic light agents to adjust the phase time and address unbalance traffic flow in different approaching lanes. The simulation results show that our Q-Learning control method outperformed the fixed-time control method.

2. PROBLEM DEFINITION

In our project, we have a road system with three horizontal roads and three vertical roads. Each horizontal road crosses each vertical road at an intersection. There are nine intersections in the road system. To simplify our problem, we assume there is only one green light in each phase. The goal of the whole program for the traffic light control system is to maximize the number of vehicles reaching its diagonal destinations per hour, without any collision and any red-light violation.

3. METHOD

This section provides background on Q-Learning algorithm, agent design, the approaches to verify the red-light violations and collisions, and the method to evaluate the throughput.

3.1. Q-Learning for traffic light control

Q-Learning is one of reinforcement learning algorithm which is model-free. Therefore, it is more suitable to handle dynamic environment compared to fixed-time control method. Each agent defines different states according to the surrounding traffic condition and chooses actions in order to maximize the reward in the long term and improve the traffic condition based on the current state. To simplify our algorithm, the communication between the agents is not employed. Despite of the lack of communication, the whole traffic light system still can reach the global optimum if each agent focuses on reaching their local optimum.

3.2. Agent Design

We view each intersection as an agent. In this section, we describe the state, action and reward function.

- Definition of State

  Our state is defined for one intersection. The intersection saturation is used to reflect diverse states of road traffic. We define the approaching lanes for one intersection as the short lanes connected to this intersection which not include other intersections. The definition of the intersection saturation is the ratio of the total number of vehicles over all approaching lanes towards one intersection to the maximum available number of vehicles over all approaching lanes towards one intersection. Table 1 shows eight states applied in our project divided by different intersection saturation. Using the ratio instead of actual number can balance the difference between intersections have different number of approaching lines.

State	Intersection Saturation
0	[0, 0.1)
1	[0.1, 0.2)
2	[0.2, 0.3)
3	[0.3, 0.4)
4	[0.4, 0.5)
5	[0.5, 0.6)
6	[0.6, 0.7)

7	[0.7, 0.8)
8	>0.8

Table 1: State and Corresponding value

- Definition of Action

  In our project, we defined four actions for traffic lights control. We used different values of a to indicate these actions:
  a = 0: traffic lights keeping the current phase. The green light which is on for this intersection does not change, and it lasts for a default time, which is 12 seconds.
  a = 1: changing the light to next phase. The green light which is on for this intersection changes clockwise, and it lasts for a default time, which is 12 seconds.
  a = 2: extending the traffic lights duration. The phase of lights does not change, but the duration time for green light increases six seconds each time this action is taken.
  a = 3: shortening the traffic lights duration. The phase of lights does not change, but the duration time for green light reduces six seconds each time this action is taken.
  Each agent is set with a default traffic lights duration. The traffic lights change to next phase every traffic lights duration time and the traffic lights duration is reset to the default traffic lights duration when changing to next phase. The action extending the traffic lights duration is used for adjusting the traffic lights duration to increase the traffic flow of the lanes with green light. Shortening the traffic lights duration reduces the waiting time of vehicles on other lanes.

- Definition of Reward Function

  Our reward function is defined as the sum of the following factors. The goal of each agent is to maximize the long-term reward.

  (1) Sum of queue length over all approaching lanes

  The sum of queue length over all approaching lanes is defined as the number of vehicles waiting behind stop line to pass the intersection. The method which we use to get the sum of queue length is to check the position of each vehicle. If the vehicle is at the approaching lanes of the intersection and the velocity of the vehicle equals to zero, the queue length increases one vehicle.

  (2) Sum of updated waiting time over all approaching lanes

  The updated waiting time for vehicle *j* at time *t* accumulate if the vehicle stops and is reset to 0 if the vehicle moves.

$$W_j(t) = \begin{cases} W_j(t-1) + 1 & vehicle\ speed = 0 \\ 0 & vehicle\ speed = 30\ mph \end{cases}$$

  The method of calculating the updated waiting time for each vehicle is to store the result in a dictionary. If the vehicle is at the approaching lanes of the intersection, get the updated waiting time from the dictionary and add to the sum of updated waiting time over all approaching lanes.

  (3) Standard deviation of queue length over all approaching lanes

  The standard deviation of queue length in approaching lanes is used to model the balance condition of queue length in different approaching lanes.

  (4) Standard deviation of updated waiting time over all approaching lanes

  The standard deviation of updated waiting time in approaching lanes is used to model the balance condition of updated waiting time in different approaching lanes.

$$\text{Reward} = \text{w1} \times \sum_{i \in l} L_i + \text{w2} \times \sum_{i \in l} W_i + \text{w3} \times \sqrt{\frac{\sum_{i \in l} (L_i - \bar{L})^2}{N}} + \text{w4} \times \sqrt{\frac{\sum_{i \in l} (W_i - \bar{W})^2}{N}}$$

  L_i: Queue length on the lane i
  W_i: Updated waiting time of all vehicles on the lane i
  $\bar{L}$: Mean value of queue length over all approaching lanes
  $\bar{W}$: Mean value of updated waiting time over all approaching lanes
  N: The number of lanes

3.3. Red-Signal Violation and Collision Detection

Red-signal violation and collision are two behaviors of traffic system that we will model in our project. In order to describe collision behaviors, we need to figure out the location of vehicles. We can use a dictionary to store the position of each vehicle as key and the number of the duplicate position as item. If we find a car instance with the same position of vehicles in the dictionary, we update the corresponding number of the duplicate position in the dictionary and the total number of collisions in the traffic system. As to the method for verifying red-signal violations, we check the position of the vehicles at the intersection and corresponding traffic lights. If the vehicle is at these slots and the corresponding traffic light is red light, we put the car instance into a list. After getting the feedback from the vehicles as the environment, we check the velocity of each vehicle in the list to see if it ran a red light.

3.4. Throughput Evaluation

To calculate the number of cars which reach their correct destinations from entries, we use the position property of each car instance to trace each vehicle. Put two pairs of diagonal departures/destinations into two sets. Each car enters the traffic system is marked with departure position. If a car reaches the positions defined as destinations, check if its departure and destination are in the same set. If so, overall throughput increases.

4. EXPERIMENTAL RESULTS

We implement our project by using Python and use Pyglet library to show dynamic graphic interface. All software programs are run in Windows platform.

4.1. Experiment Setting

The environment for this project is a road system with three horizontal roads and three vertical roads as Figure 1. It contains 9 intersections. Between two intersections, there are 30 slots. During the simulation, new vehicles are generated from 4 corners. At the beginning of each time step, among all the entries have no car currently, inject one car at one entry randomly.

For information exchange between V group and I group, V and I group share a 70*70 two-dimension array to put light and car objects. For us I group, we provides light colors and their positions in a light class for V group to make a decision, and for V group, we get information of cars from car class, such as position and velocity. V group uses intersection list to receive information about lights, and I group uses car list to get information about cars.

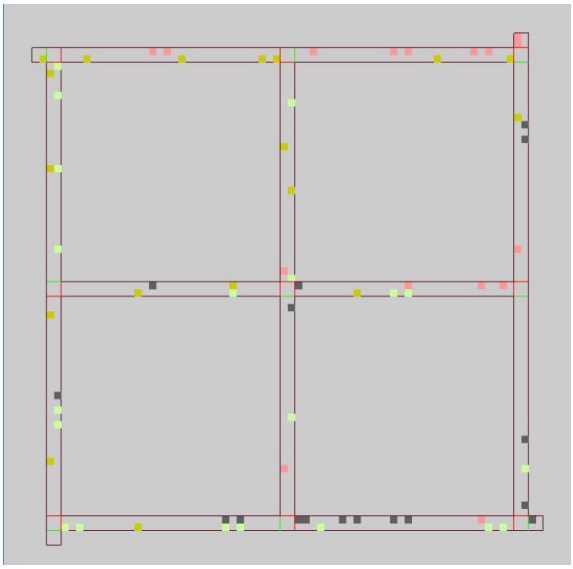

Figure 1: Environment for the simulation

4.2. Parameter Setting

We did some trials on the parameter selection within reasonable regions. Figure 2,3 show the throughput of different epsilon, learning rate and discount factor as the simulation time increases.With combination of ϵ for exploration=0.2, learning rate α =0.5, discount factor γ =0.7, the throughput reaches maximum.

For reward coefficients, we found that setting standard deviation of queue length over all approaching lanes and standard deviation of updated waiting time over all approaching lanes of larger absolute value than sum of queue length over all approaching lanes and sum of updated waiting time over all approaching lanes is an optimal choice. This indicates that standard deviation of queue length over all approaching lanes and standard deviation of updated waiting time over all approaching lanes contribute more when calculating reward.

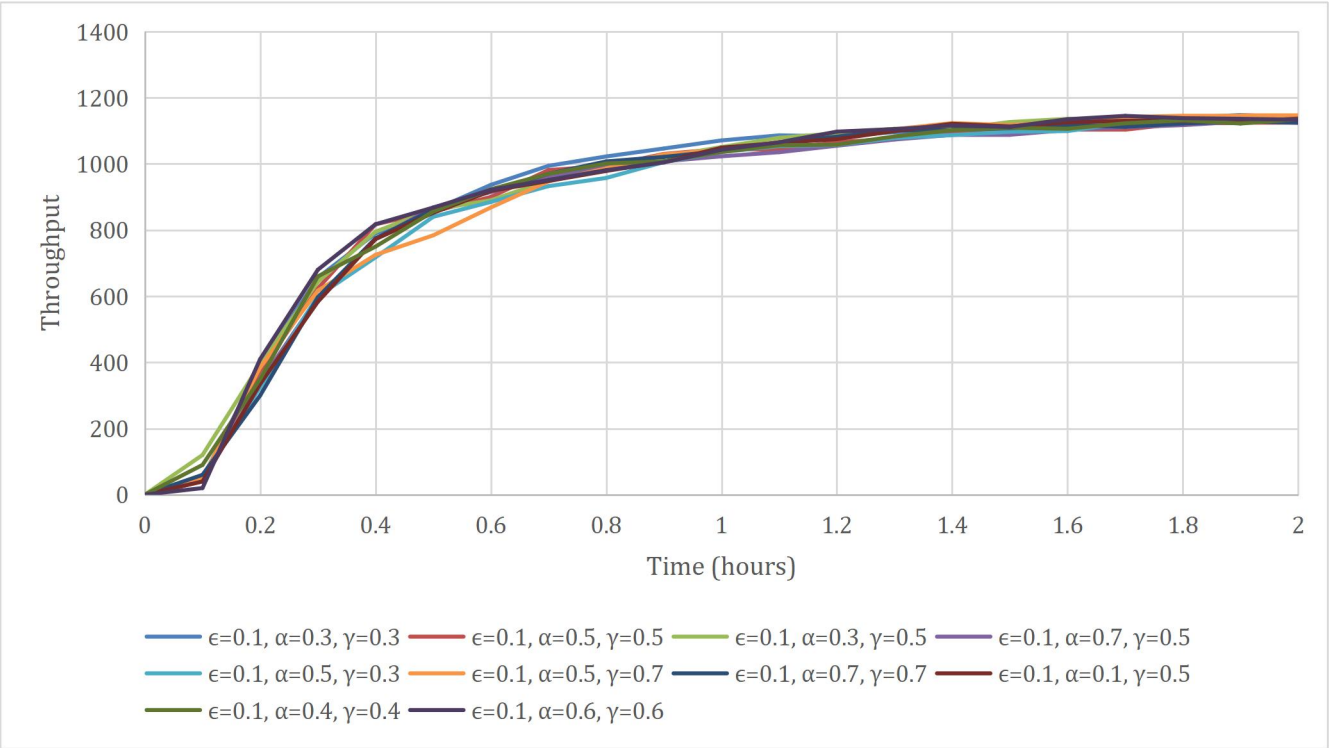

Figure 2: Throughput with different parameter settings when ϵ=0.1

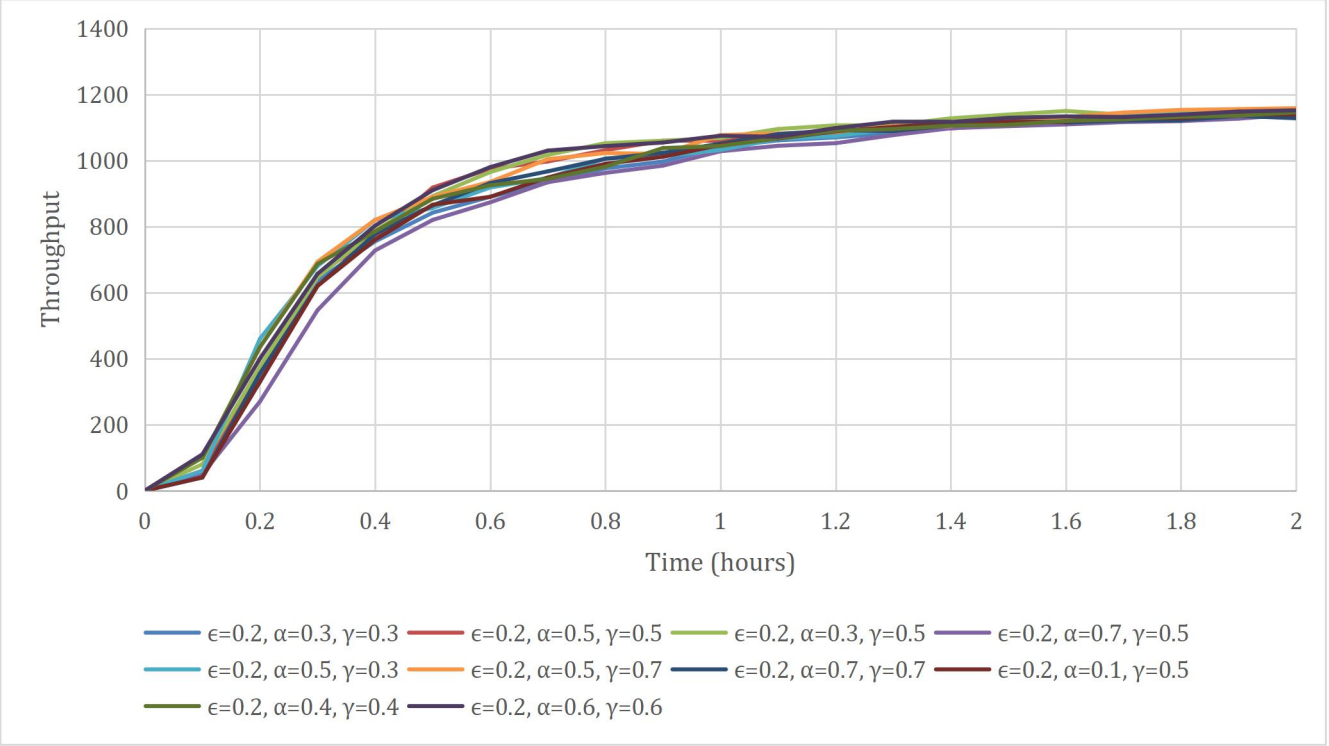

Figure 3: Throughput with different parameter settings when ϵ=0.2

In table 3 and 4, the parameter setting and reward coefficients for our method are shown.

Parameter	Value
Action time interval Δt	2 seconds
Discount factor for future reward γ	0.7
ϵ for exploration	0.2
Learning rate α	0.5
Default traffic lights duration for each phase	12 seconds

Table 3: Setting for our method

w1	w2	w3	w4
-0.25	-0.25	-0.55	-0.55

Table 4: Reward coefficients

4.3. Compared Methods

To evaluate the effectiveness of our Q-Learning control method, we compare our method with the conventional method for traffic light control.

- Fixed-time Control
  Fixed-time control method is a common method for traffic light control. The phase changes to next phase according to pre-determined traffic lights duration. However, it method is hard to address dynamic environment because of the variation of the traffic flow.

4.4. Performance

To estimate the performance of our traffic light control system, our system is evaluated by overall throughput of the road system. The throughput is the number of vehicles that reach the destinations per hour. The higher overall throughput achieves, the better the performance is.

According to our simulation results, the maximum throughput of Q-learning control method is 11.2% better than the throughput of fixed-time control method.

As simulation time increases, throughput improvement becomes saturate. Define throughput converges to time t, when its enhancement towards maximum throughput is less than 1%. For maximum throughput using Q-Learning control method, convergence time is 1.7 hours. For maximum throughput using Fixed-time control method, convergence time is 1.8 hours.

There is no collisions and red-light violations.

Methods	Maximum Throughput	Convergence Time
Fixed-time control method	1026	1.8 hours
Q-Learning control method	1141	1.7 hours

Table 5: Performance of fixed-time control method and Q-Learning control method

The number of collisions	The number of red-light violations
0	0

Table 6: Collisions and Red-light violations

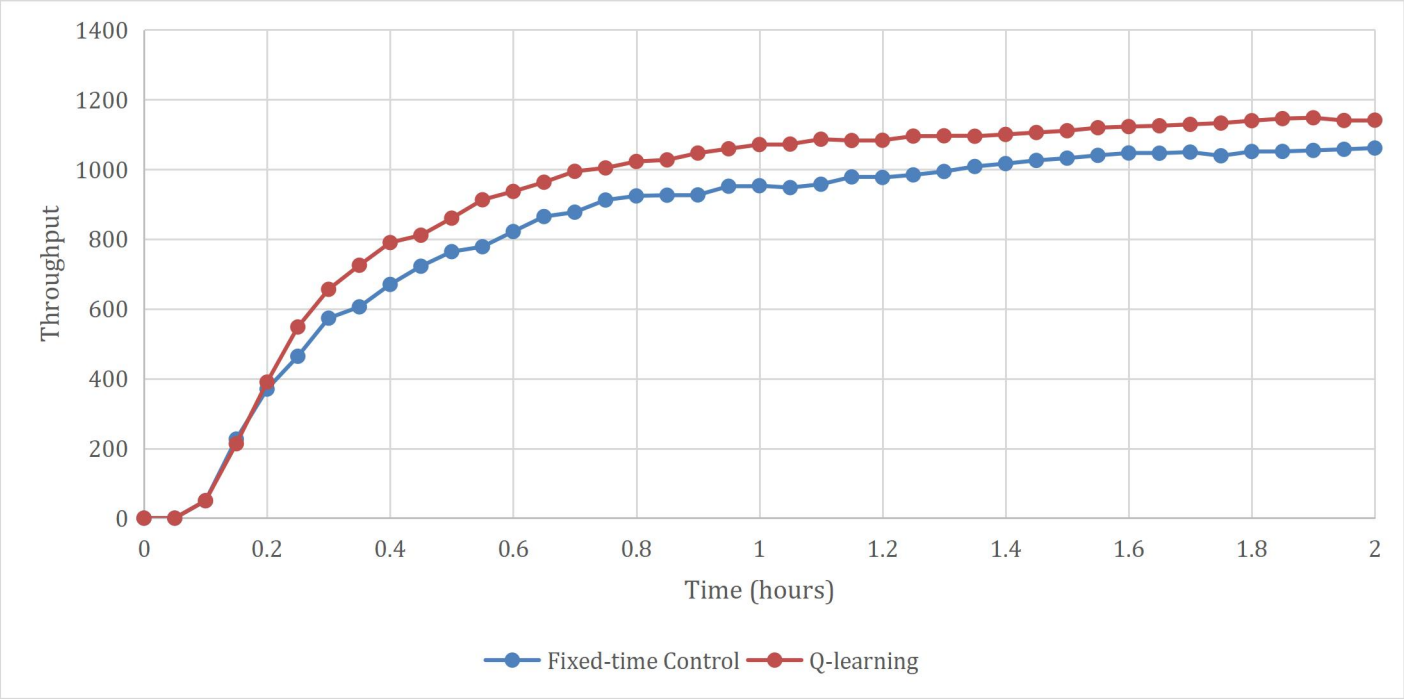

Figure 4: Throughput with Fixed-Time Control and Q-learning