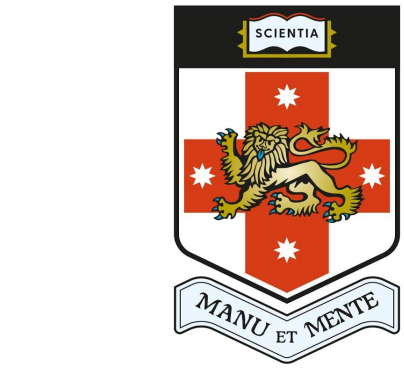
# Reinforcement Learning

COMP9417, Assignment 2




UNSW
SYDNEY

**z5104857 Xiaoyun Shi**
**z5102866 Jinzhu Wu**
**z5100331 Li Chen**
**z5092923 Jintao Wang**

# 1 Introduction

The primary function of the traffic signals as we know is to assign the right of way to the contradicting movements of traffic in an intersection. This is actually done by allowing the conflicting traffic streams to share the same intersection by way of separating the time. Furthermore, the signals provide an orderly movement of the conflicting flows by means of assigning the right of way alternately to various movements of the traffic. However, it may interrupt extremely heavy flows to allow the crossing of some minor movements which otherwise, could not move safely into the intersection.

When they are properly timed, traffic signals may increase the traffic handling capacity in an intersection. Additionally, particularly, traffic signals may also help in reducing various types of accidents. However, as with everything, there are disadvantages too. Traffic signals are not really a solution to all problems in road intersections. An unwarranted signal can also adversely affect the safety and the efficiency of the traffic. For example, there is an excessive delay because of the time allocated in the traffic signals.

The aim of this project is to implement a set of traffic lights designed to reduce the delay at lights. The basic purpose for this task is:

1. Simulates the traffic generation, flow and traffic lights
2. Displays the simulation on the screen
3. Controls the lights using a fixed change time of 10 time-steps
4. Uses a reinforcement learning algorithm to improve on 3.

To achieve this purpose, we use several learning methods and also implement some extensions as suggested.

# 2 Method

These project mainly use two reinforcement learning technique: Q-learning and Sarsa. The main difference between this two is that Q-learning updated according to the maximum value(somewhat like to take the ‘maximum’ action), and have no relationship with the current strategy as well. However, the next action of Sarsa is

decided when it take the current action(on-policy algorithm), which is the difference between these two methods.

## 2.1 Q-learning

The main method we use is Q-learning. Q-learning is a reinforcement learning technique used in machine learning. *Q*-learning can handle problems with stochastic transitions and rewards, without requiring adaptations[1].

For any finite Markov decision process (FMDP), *Q*-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable.[2] *Q*-learning can identify an optimal action-selection policy for any given FMDP.

The main algorithms is 'Q(s,a)=r+γ(max(Q(s′,a′))'[3] and epsilon greedy means the percentage of the decision. if epsilon greedy equals to 0.9 e.g, which means it has 90% possibility to do the choice as the best solution of the Q table and there are 10% possibility to do random action. alpha is learning efficiency, it will determines the how much errors of it should be studied and alpha should smaller than 1. Furthermore, gamma is attenuation value, if gamma is 1, we can predict the further rewards of this choice, however, if gamma equals to 0, we can only know the current rewards of our choice. Therefore, the range of gamma from 0-1 means the choice made would be generally more wise.

## 2.2 Sarsa

The Sarsa algorithm is an On-Policy algorithm for TD-Learning. The major difference between it and Q-Learning, is that the maximum reward for the next state is not necessarily used for updating the Q-values[4]. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. The name Sarsa actually comes from the fact that the updates are done using the quintuple Q(s, a, r, s_, a_)**.** Where: s, a are the original state and action, r is the reward observed in the following state and s_, a_ are the new state-action pair[5]. Unlike Q-learning, Sarsa will consider next action and will not be implemented in while loop.

Lambda is range 0-1( like gamma) which is  a root used to record the state we have already achieved,  if lambda is 0 we can  only update one step, however if lambda nearly equal to 1, we can round update, which largely improve the update efficiently.

# 3 Extension

The first extension is based on the traffic light, the basic algorithm of the traffic light only contain red light and green light, which cannot satisfy the requirement of this project, because it would cause car crash. In order to solve this problem and reduce the delay of the light, we add the amber light to the existing light. This light is used as a transmitting light between red to green, when this light is in, cars are not allowed to cross the intersection, therefore, it solves the problem of car crush.

The second extension is the improve the calculation ways of car positions about the cars which runs in opposite direction. This method ensures us know which cars position is smaller and then know how to handle with this situation properly.

The third extension is test different traffic intensities and generating functions. The basic method to get the traffic intensities is derived from the ‘time% (rnd.nextInt(10 ) + 5) == 0’,  the parameters in ‘rnd.nextInt’ can be used or modified to control the traffic intensity, Therefore, we will try different parameters to do multiple experiments about the effects of different traffic intensities.

The fourth extension is vary RL parameters. These parameters dominate the effect of the reinforcement techniques we used in this projects. The default parameters are respectively discount factor(gamma = 0.9), learning rate(alpha = 0.1) and Epsilon greedy exploration(10%). Our work is to get the different results by changing these parameters.

The fifth extension is try different reward functions eg -1 per car on the grid. The basic method is to decrease the reward by one once find a car stop(no matter how much cars stop in this road, zero otherwise). Our new modification is to accumulate the how much car stop in one road(1 car found the reward would -1), which is different from the original method.

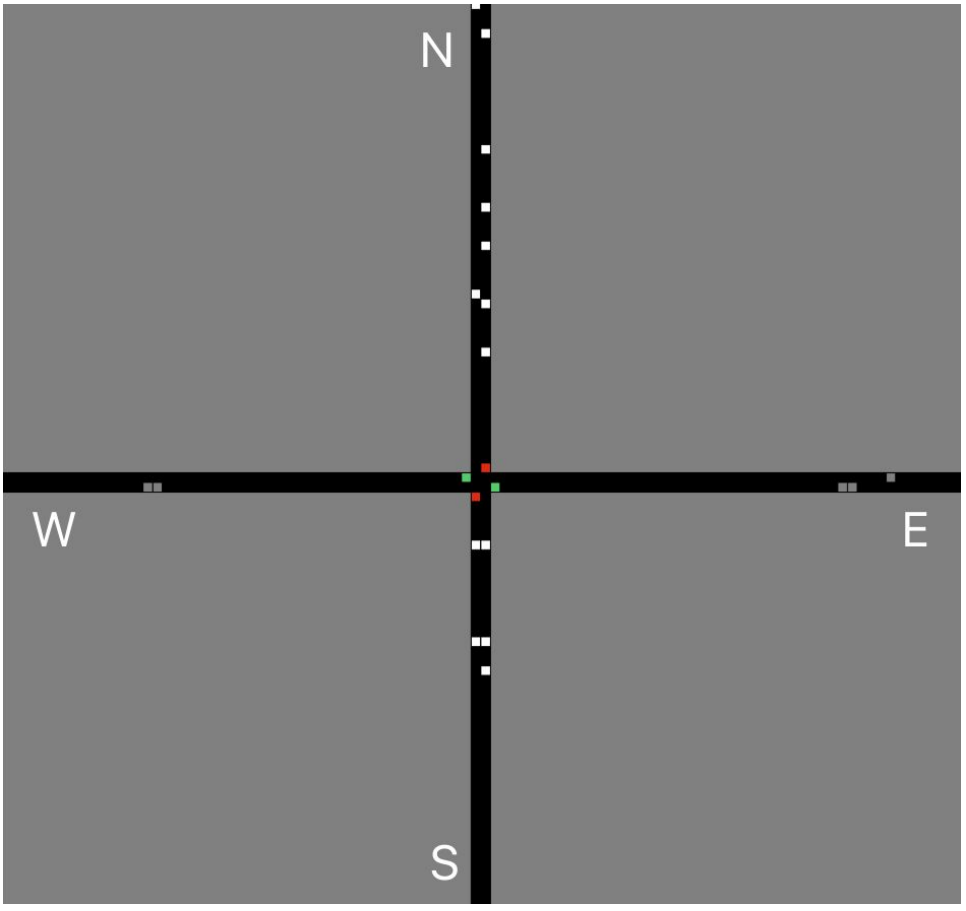
The sixth extension is try different exploration strategies. We attempt to try two reinforcement techniques respectively ‘Q-learning’ and ‘Sarsa’. There are some difference in the action choose of these two methods. Q-learning will not know the next action, however, Sarsa will know the next action.

# 4 Environment simulator

The extensions for the traffic model we have tried to implement are ‘Include amber phase in lights’ and ‘Include 2-way traffic’.

## 4.1 Traffic model establishment

To start with, traffic is flowing two ways on two intersecting roads, each road have two converse directions. Therefore, the whole vertical roads are composed by four one-direction paths. Cars can only travel in West-East, East-West, North-South and South-North directions. The road length is 100 units.




https://youtu.be/rewWYDvclsc

The crossroads is in the middle of two roads, and the traffic lights are located around the intersections.The traffic is controlled by four traffic lights, which are actually two

sets of traffic lights. Each set controls the access of one road. And there are three different signals of lights, which are RED, AMBER and GREEN. If a car is stopped by a Red or Amber light, it cannot move forward. One unit of road holds only one car, the cars cannot occupy the same space, it means that other cars need to queue behind stopped cars. A controller can choose to change the access state by 'Switch' or 'Not Switch', but only after waiting at least three time-steps since the last change. If the light switch from Green to Red, it will have a intermediate signal of Amber. Cars enter the road some way from the intersection at random intervals, the simulator moves the cars one space along the road at each time-step and cars are removed from the road when they reach the boundary of the simulated region.

## 4.2 Implementation

We firstly import tkinter module to get a visible window to simulate the intersecting roads, cars and traffic lights, in order that we can see what is happening when cars are running. At the beginning, we set up the simulation with a base case 10 time-steps switching, then use a reinforcement learning algorithm to improve it. The learning process takes place according to the observation and actions. At each time-step, an action is selected by the learn function which finally return a optimal choice or random choice by epsilon-greedy exploration.

More detail for the program have written in the README.md file.

# 5 Analysis

## 5.1 Try different exploration strategies

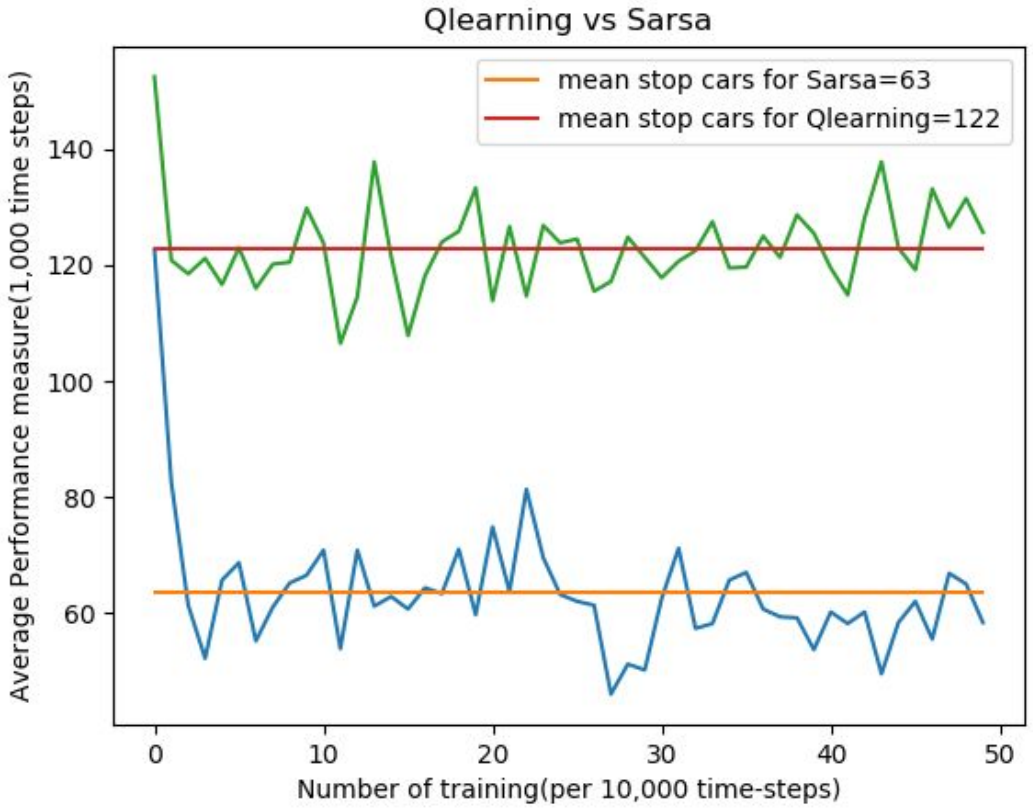



Figure 5.1.1

The Figure 5.1.1 shows the different performance between two exploration strategies. Average performance measure stands for the sum of the number of cars is queued in a period of time(per 1000 time-steps). We use default parameters, reward: -1 each road, discount factor: gamma = 0.9, learning rate: alpha = 0.1, epsilon-greedy exploration 10%, generate cars: time % (rnd.randint(1, 10) + 5) == 0 and the prob of generating cars on each road is fair(25%, 25%, 25%, 25%). By comparing Q-learning and Sarsa under the default parameters, Sarsa performs better than Q-learning in reducing the number of stop cars. Therefore, we will do further comparative analysis by Sarsa algorithm.

## 5.2 Try different reward functions eg -1 per car on the grid

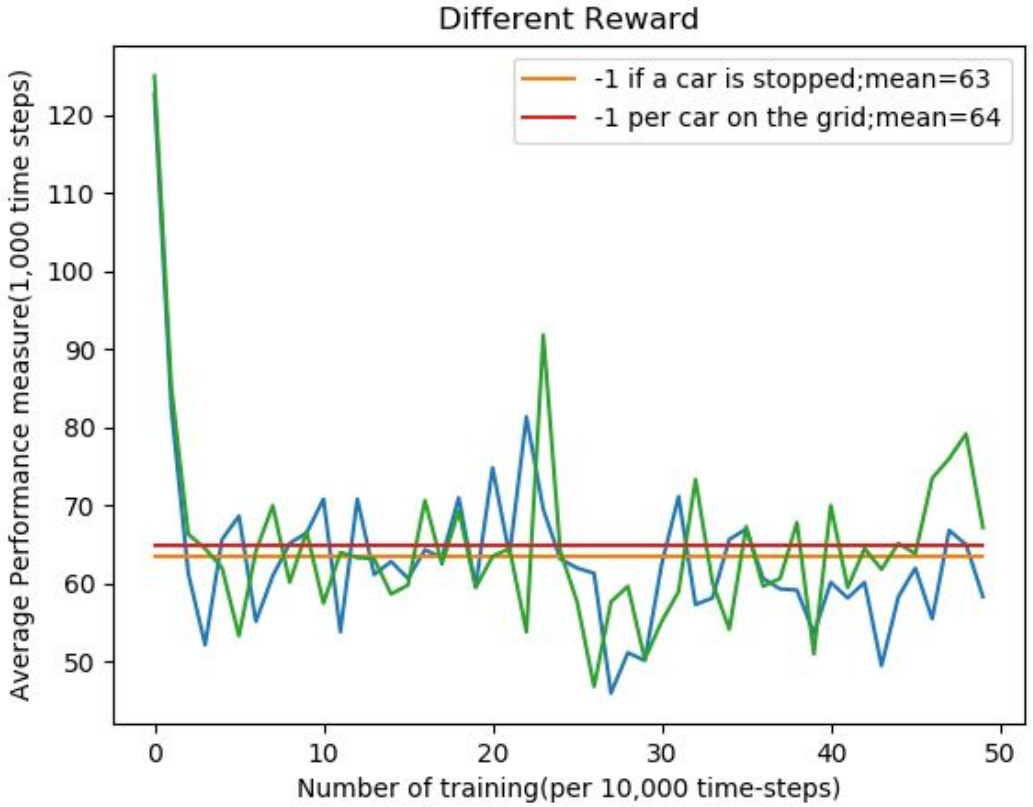



Figure 5.2.1

The graph figure 5.2.1 shows the information performance measure of Sarsa in two different reward calculations ways. The parameters exclude the reward are same(Discount factor: gamma = 0.9 , Learning rate: alpha = 0.1, Epsilon-greedy exploration 10%,Generate cars: time % (rnd.randint(1, 10) + 5) == 0). The reward of Sarsa is reward -1 once we find a car stop, the reward of Sarsa1 is reward -1 per car find on the grid.

It is clear that there is no large gap between the average performance measures of those two reward functions(respectively 63 and 64), which makes us think that if our traffic intensity are too small, it means that small traffic intensity cannot reflect the impact of different reward approaches, because it is unlikely that there are more than 1 car stop on one road at the same time when the traffic intensity is small. Therefore, there are may not have large difference in the reward of these two reward way.

In conclusion, we think the car intensity largely affects the reward so that it can not show the difference between those two different reward ways. Therefore, we decide to take another test, in this test we will use large traffic intensity to make the possibility of over 1 car stop in one road largely improved.

## 5.3 Test different traffic intensities and generating functions

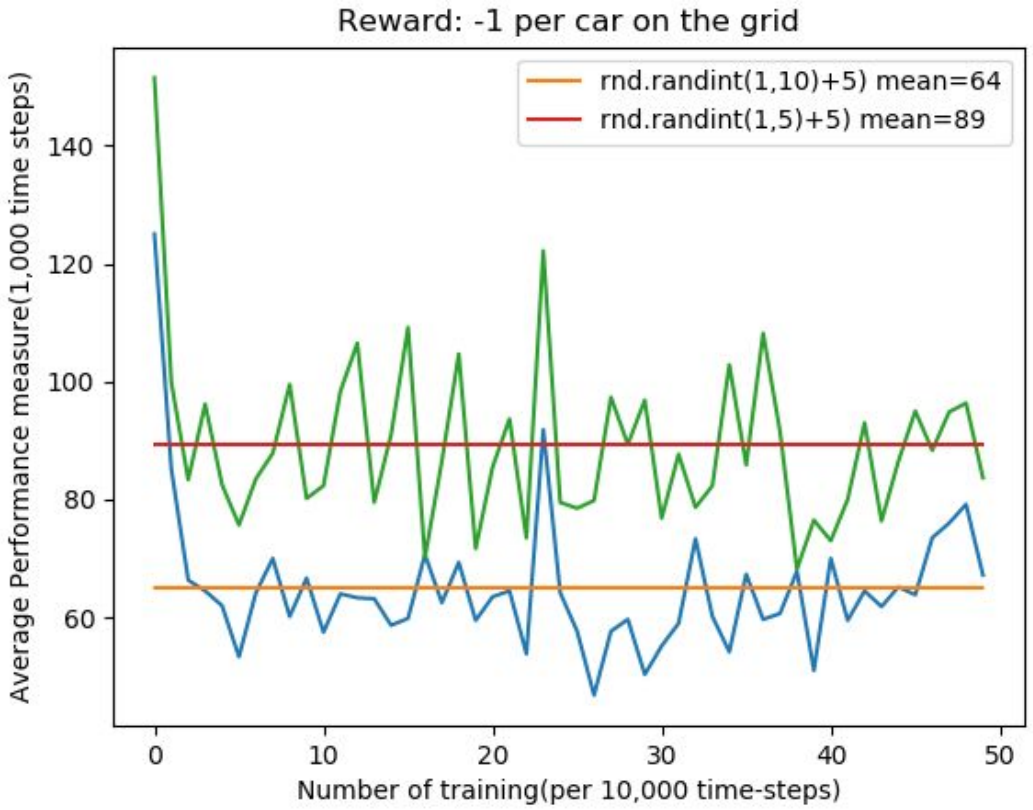



Figure 5.3.1

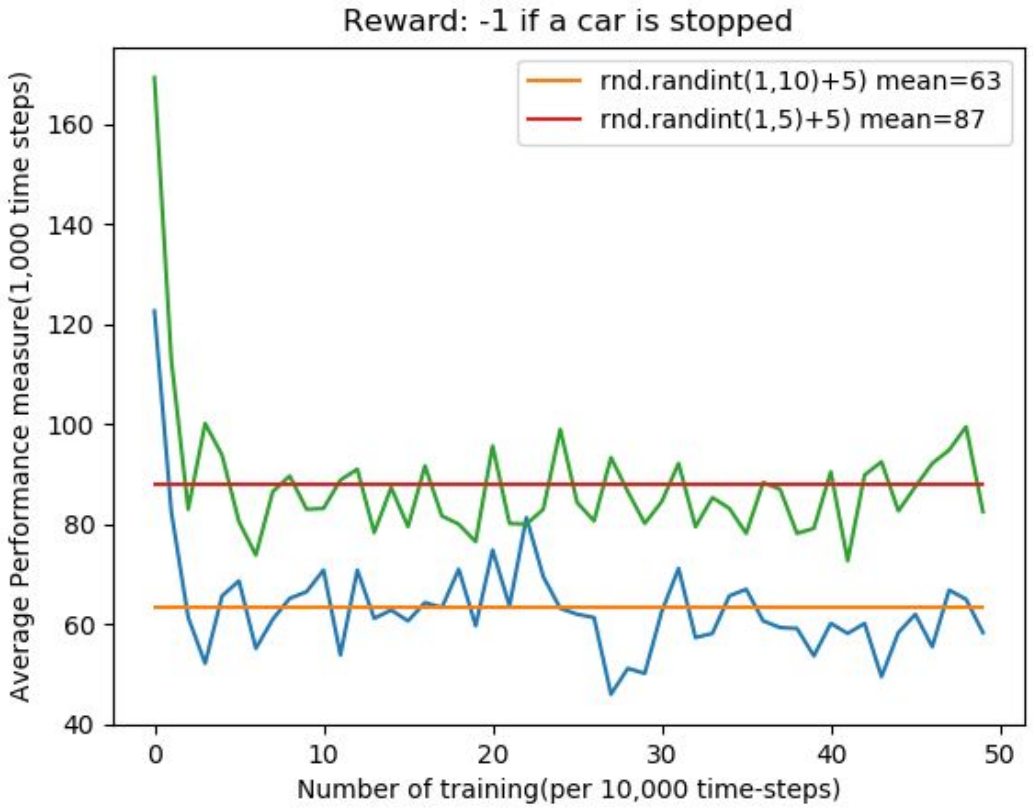



Figure 5.3.2

There are two figures(figure 5.3.1 and figure 5.3.2), they have same same parameters exclude the reward calculations ways and car generating formula(Discount factor: gamma = 0.9 , Learning rate: alpha = 0.1, Epsilon-greedy exploration 10%).

The figure 5.3.1 calculate the reward -1 per car it find stop and it also has two ways of generating cars(respectively  time % (rnd.randint(1, 10) + 5) == 0 and Generate cars: time % (rnd.randint(1, 5) + 5) == 0). We can find that in two transport intensity, the mean of performance measure are quite different, respectively 64 and 89.

The figure 5.3.2 has same two car generating way, but the reward calculation way is different, it will reward -1 if find a car stop and we can find the mean of these two lines do not have a large difference with figure 5.3.1, respectively 63 and 87. Moreover, the lines of the figure 5.3.2 are more stable and has less fluctuations than figure 5.3.1.

In conclusion, we can derive that the default parameters of reward calculation way is much better than the one we try to use.

## 5.4 Vary RL parameters

The following three figures respectively using different learning rate, different discount factor and different epsilon-greedy exploration to analyze vary parameters' effects to performance measure. We all set the same reward( -1.0 if a car is stopped at a red light on either road, including the case that the light delay is less than 3 time step, 0 otherwise.) and same the prob of generating cars: time % (rnd.randint(1, 10) + 5) == 0.

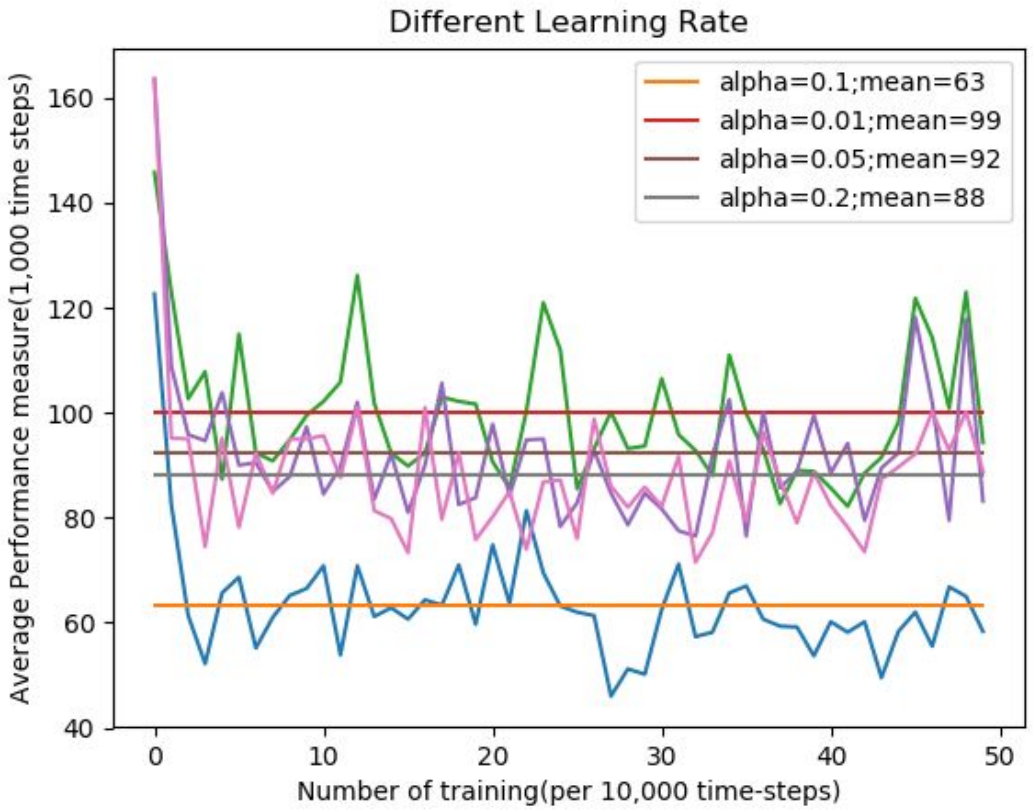



Figure 5.4.1 Different Learning Rate

In Figure 5.4.1, when alpha= 0.01, the learning gain is very little, this shows that a low learning rate perform not well in learning processing. And we can also find that the performance do not have a positive correlation with learning rate. Increasing the learning rate will not necessarily improve performance. A suitable learning rate can minimize the number of delay for cars. In our cases, a learning rate with alpha= 0.1 does best in performance measure.

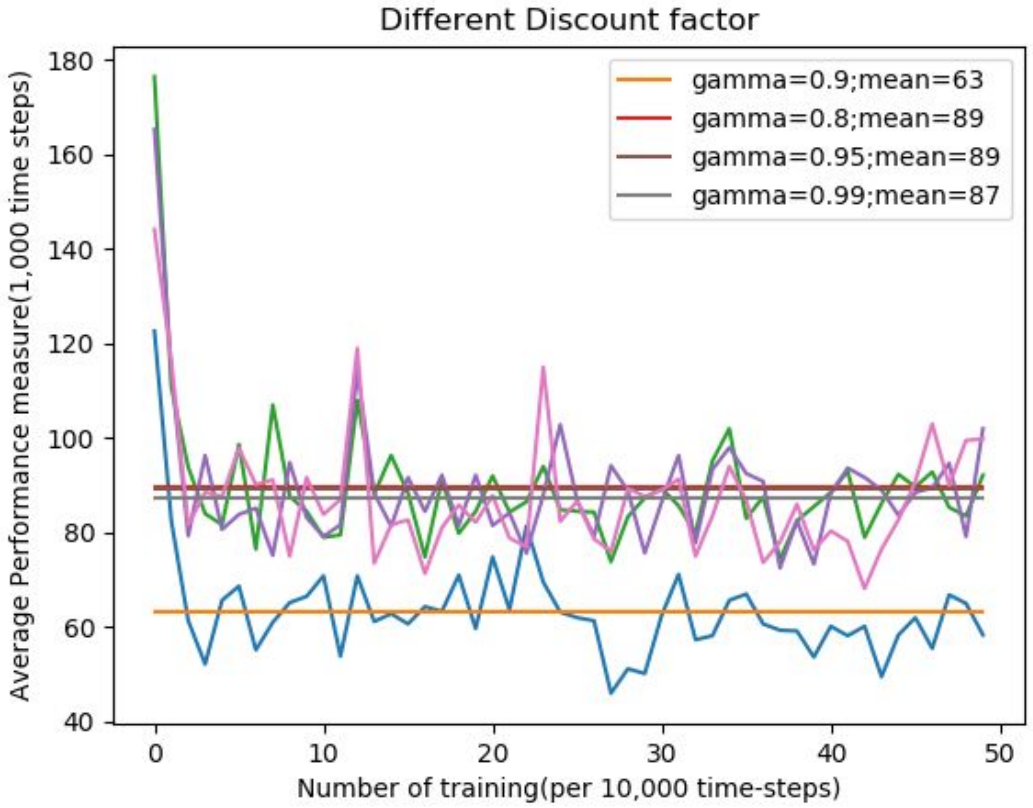



Figure 5.4.2 Different Discount factor

In Figure 5.4.2, discount factor have similar effect in performance measure. The discount factor gamma is used to adjust the significance of future rewards. When we set gamma to 0.8, 0.95 and 0.99, the delay of cars can be reduced, but a gamma value of 0.9 obviously do better than those three. So we can also surmise that the performance do not have a positive correlation with discount factor.

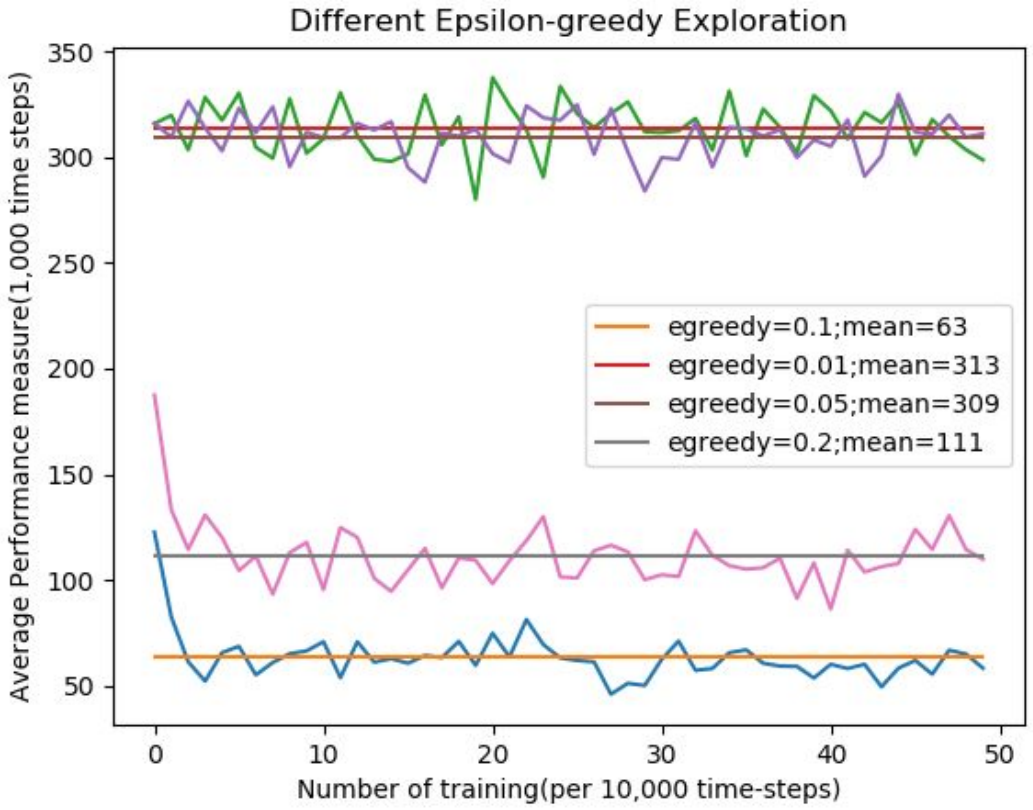



Figure 5.4.3 Different Epsilon-greedy Exploration

In Figure 5.4.3, different epsilon-greedy explorations have huge effect in performance measure. When the proportion of greed is very low, the selection process tends to be random, so that it seldom learns optimal options. But when the proportion of greed is high, the selection process tends to choose the best way that it thinks, so that it rarely try to finds new options which maybe better than previous choice.

# 6 Related work

Nowadays, people are tend to pay more attention to the structure of transport system, how to improve it by using  new algorithms such as reinforcement techniques such as Q-learning and Sarsa. By the way, even though people know these methods may have positive effect in the research, they are still struggling in find the appropriate way to use them in the reality word. By the way, there are many researchers have examined that Q-learning really would encourage the development of the adaptive traffic control. The effect of Q-learning are extremely useful in the

control of the traffic light, this is also examined by the project we clarified in this report. Therefore, the difficulty of these algorithms is that whether it can work well in real world (high traffic intensity) as it in the research.

In conclusion, it is not easy to implement the same system in real word, because it will need not only careful placement of different structure and appropriate parameters, it also costs a lot to actually obtain some sensors which is a important part of the whole plan. Besides, it's not the end of the difficulty, these applications also need time, money and people to maintain it works well.

# 7 Conclusions

In conclusion, we can find that the default parameter is the best one in our cases. In the experiments we have already done, compared with other parameters which we modified, the default parameters always perform better than others. We try to change the gamma, alpha and greedy to at least three different numbers, but only to find they perform worse than before. As we can know, if the greedy too large, it will keep find the best route it thought instead of finding a new route, therefore, it cannot find a better route. Moreover, if the Epsilon-greedy Exploration is too small, the random rate would be extremely high, the results would derived from the random answers, therefore, it can not achieve to the best state as well. When it comes to other parameters we can find if we change the default values, the line would turn to have a large number of fluctuations and we cannot find a better improvement in performance measure than the default one.

# References

[1] *Reinforcement Learning: An Introduction. Richard Sutton and Andrew Barto*. MIT Press, 1998.
[2] Francisco S. Melo, "Convergence of Q-learning: a simple proof"
[3] Matiisen, Tambet (December 19, 2015). "*Demystifying Deep Reinforcement Learning | Computational Neuroscience Lab*". *neuro.cs.ut.ee*. Retrieved 2018-04-06.
[4] F. Bertoluzzo and M. Corazza: *Reinforcement Learning for automated financial trading: Basics and applications*. In: S. Bassis, A. Esposito and F.C. Morabito (Eds.): *Recent Advances of Neural Network Models and Applications, Springer*, 197-213, 2014
[5] C. Gold: *FX Trading via Recurrent Reinforcement Learning. IEEE International Conference on Computational Intelligence in Financial Engineering*, 2003.