Control Method of Traffic Signal Lights Based On DDPG Reinforcement Learning

Haosheng Wu
School of Software Engineering, Chengdu University of Information Technology, Chengdu, Sichuan, 610225, China
*Corresponding author’s e-mail: 1561752926@qq.com

Abstract. Optimizing the traffic route in the current environment and how to control the best time for traffic lights in large-scale traffic flow has become the main factor in people’s current traffic travel life. This article proposes the use of deep deterministic policy gradients (DDPG). DDPG reinforces the method of learning traffic lights to control the best time for traffic lights, and studies the information interaction between the agent in the environment using the intersection as the agent, so that the agent can find the controllable target in the shortest time. The comparison of simulation experiments with traditional neural networks and reinforcement learning algorithms shows that the algorithm proposed in this paper is superior to other traditional algorithms in terms of solution time, and can quickly and effectively control the timing of traffic lights.

1. Introduction
With the development of technology and economy in today’s society, people’s lives have gradually improved, and cars have gradually become a means of transportation for humans. This has led to traffic congestion becoming a more serious traffic problem in the current society. For example: waiting time is too long, lanes are occupied The rate is too high and so on.

This paper proposes a method based on deep reinforcement learning, which can greatly improve the control strategy of traffic lights by implementing deep reinforcement learning algorithms. The reinforcement learning algorithm used is to learn the agent by taking the traffic intersection as the agent and through the continuous trial and error method. Corresponding actions, in a given traffic intersection, by calling the algorithm to change the timing of the signal lights and change the code of conduct for certain intersections, for example, at certain moments, we can appropriately extend the time of turning left and going straight traffic lights, Optimize traffic flow through corresponding changes. This article uses a single agent as the main application method, taking the intersection as the single agent. The agent obtains the observation value of each intersection by observing and inspecting the four intersections in the south, east, north and west, and performs the observation of each intersection through the observation value, judgment. Although the reinforcement learning of a single agent is performed in a simple crossroad environment, the following difficulties will be encountered when designing the algorithm model:

- When the environment of the agent is relatively wide, the agent will explore the surrounding environment, and will spend a lot of time in the exploration time, and will spend a lot of time in the learning and training of the agent.

- Agents are in the same environment to achieve the maximum expected return. How to make the expected value of our needs converge optimally and quickly under the premise of a large amount of data, without excessively wasting computing resources.
In response to the above problems, I proposed a general single agent learning algorithm based on the DDPG algorithm. The main contents of this calculation are as follows:

- Joining the memory experience pool, in different training stages, will store a series of experiences of the previous stage, and will refer to and advance the experience in the new stage of learning.

Based on the DDPG algorithm, this article improves the reinforcement learning of the agent so that the agent can perform calculations quickly and stably.

2. Related Work
Deep reinforcement learning is currently a popular method in artificial intelligence methods. In 1954, Minsky first proposed the concepts and terms of "reinforcement" and "reinforcement learning". With the trend of history, reinforcement learning was gradually clarified and was finally designated as "The concept of "trial and error" is now the core part of intensive learning[1][2]. In 2015, the AlphaGo program developed by Google DeepMind defeated advanced human Go players. AlphaGo uses deep reinforcement learning to continuously adjust the strategy during continuous trial and error training to maximize the cumulative reporting reward[3].

2.1. Single Agent Reinforcement Learning And Its Development
In 2013, DeepMind published a paper using reinforcement learning to play Atari games, opening a new chapter of reinforcement learning[4]. At that time, people did not use deep reinforcement learning too much to apply deep reinforcement learning to other fields because of the deep reinforcement learning at the time. The algorithm is not stable. In 2015, Mnih V et al. combined convolutional neural network and reinforcement learning and proposed the DQN algorithm[5]. The DQN algorithm uses the memory tour technology in the early reinforcement learning methods to perform efficient learning through learning from the previous learning experience, breaking the deadlock of the deep reinforcement learning algorithm at that time.

2.2. Single Agent Reinforcement Learning Algorithm
The most typical of traditional single-agent reinforcement learning algorithms is the environment of Markov decision process. Through the observation of the agent, it can judge the state \( s_t \) of the surrounding environment at different time \( t \) in time, and the environment state is defined as a set \( S \), And \( s_t \in S \). At the same time \( t \), the agent obtains the corresponding observation value \( o_t \) in this state by initializing the observation function. According to the observation value, the agent can use the corresponding learning policy to give the agent the corresponding action selection in the current state at \( t \). In the same way, we define the observation value \( O \) and the action selection \( A \) in the process as a set, And \( o_t \in O, a_t \in A \), it can be seen that the choice of action is based on the choice of actions on the strategy \( \pi \) through observations, so \( a_t \sim \pi(o_t) \), the agent will get after the corresponding action is selected The corresponding reward \( r_t \) at time \( t \). After the process is over, update the current state \( S_{t+1} \) and start the next step of learning. And start the next step of learning, the algorithm framework is as follows:

![Figure 1. Reinforcement learning algorithm framework](image-url)
3. Design Description of DDPG Algorithm

The full name of DDPG is Deep Deterministic Policy Gradient. First of all, this algorithm is similar to the previous DQN algorithm and uses an experience pool and a dual network structure to enable the neural network in the probability algorithm to learn with maximum efficiency. In other words, the ultimate goal of the DDPG reinforcement learning algorithm is to allow the agent to choose the optimal action in the learning environment. How to choose the optimal action is to obtain an optimal behavior strategy through the observations of the environment. The purpose of the strategy is to allow our agent to obtain the largest reward in a given environment. DDPG can find the optimal behavior strategy in the given Markov decision to maximize the future cumulative reward.

In this paper, the intersection in the environment is used as an agent, and set according to the requirements of the enhanced algorithm in the second section. Set the road section around the intersection as the environment and set the state to $s_t$ according to the environment. Make a decision and choose behavior $a_t$ according to the current state to complete the decision. After the action, according to the reward $r_t$, the algorithm framework of DDPG is shown in the figure below.

**Figure 2. DDPG algorithm framework**

In Figure 2, it can be found that the Actor network chooses behavior according to the strategy, and the Critic network continues to get the corresponding reward under the premise of the strategy according to the action taken by the agent. The specific formula is as follows:

$$Q^\pi(s_t, a_t) = E[\lambda Q^\pi(s_{t+1}, \pi(s_{t+1})) + r(s_t, a_t)]$$

According to the current actions, status and rewards obtained by the agent, it can be regarded as the previous accumulated experience and put it in the memory experience cache. After a certain step is executed, a part of the experience is obtained from the memory experience and used in the DDPG network of the agent. training. In the end, the difference between the actual reward obtained by the agent and the previously expected reward is required to obtain the loss value $L$.
Relative to the reward function \( R \), we will perform a reward and punishment method on the agent \( a_t \) at the corresponding time \( t \). After each execution of the corresponding behavior, rewards and punishments are required. The agent will be rewarded using the following formula:

\[
L = \frac{1}{N} \sum_{i} \left( r_i - Q(s_i, a_i | \theta^0) \right)^2
\]

The report for \( r \) at time \( t \) is shown in the figure above, where \( e \) is the total number of road segments in the model road, \( h_i' \) is the occupied length of vehicles on the road section at time \( t \), \( l_i' \) is the length of the road section at time \( t \), \( N_i' \) is the total number of vehicles at this moment, at each time step. With the implementation of the corresponding learning progress, the vehicle's occupation rate on the road will become smaller and smaller, so the return function here is in a decreasing form. It is more intuitive to use negative numbers to express, and the reward of the agent will be executed, and finally The expected reward function \( R \):

\[
\mathbb{E} Q_{\theta}(s, a) = \mathbb{E} [r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots | s_t = S, a_t = A, \Pi]
\]

At this point, you can think about the ultimate goal of the agent is to reduce the waiting time of the vehicle at the intersection. In the later long-term learning, it is necessary to find an optimal strategy to maximize the accumulated reward.

The function \( J \) is commonly used in DDPG to measure the quality of a strategy, where \( N \) is the number of experiences in the memory cache:

\[
\nabla_{\theta'} J(\pi) = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta'} Q(s_i, a_i | \theta^0) \pi(s_i | \theta^0) \nabla_{\theta'} \pi(s_i | \theta^0)
\]

4. Evaluation

4.1. Lab Environment

For the authenticity of the design, this paper uses SUMO to simulate the traffic environment, defines 5000 cars, starts the traffic signal simulation experiment of traffic simulation, and also uses the other 2 algorithms to compare with the DDPG algorithm of this group, respectively. With NAF algorithm and DQN algorithm, SUMO is an open source road model, it can meet the relevant data collection required in the simulation experiment, as well as the simulation of traffic behavior and the required road network construction. The most important thing is that it can also collect the timing data of traffic lights. The above extensions need to be improved.

4.2. Experimental Environment Settings

In the course of the experiment, we conducted a total of 1 experiment. The experiment was conducted under a dual-access intersection. First, without adding any reinforcement learning algorithm to the original intersection, observe the total waiting time for vehicles passing the vehicle during this period and the final reward value. Intersection two-way lane environment, as shown in the picture:
4.3. Experimental Results and Analysis

We used the DDPG, DQN, and NAF algorithms for the above experimental environment. These three types of algorithms all contain empirical buffer pools, so for the same performance, we set them to 3000 and the minibatch size to 32. Put the above algorithm in the above SUMO environment for learning until the final target value we want to obtain converges. In this chapter, I will use the number of steps per round and obtain the corresponding reward for evaluation, and also evaluate based on the total waiting time of each round. In order to make the data more accurate, the result of multiple training and averaging is carried out on the rewards used.

Figure 6 is the results of the DDPG algorithm and NAF algorithm in the intersection environment. The results obtained according to the waiting time of the vehicle. During training, the training and exploration time of NAF and DDPG is relatively short. On the contrary, DQN consumes a long time on this algorithm. Compared with DDPG, the convergence speed of NAF in the first half is faster than that of DDPG, but in the second half, the convergence of DDPG gradually exceeds NAF. When comparing the total waiting time of vehicles, without adding any algorithm, it takes 338,798 steps to complete a round in SUMO environment. Figure 6 shows the improved results, because the reward results are obvious. It can be seen that the convergence speed of DQN is slow, so we only compare the DDPG and NAF algorithms.

![Figure 4. Environment test](image-url)
5. Conclusion
In this paper, the deep reinforcement learning method and the intelligent body are combined and applied to the SUMO traffic simulator, which is a good realization of the control of traffic lights to improve traffic jams. In the algorithm species, the experience buffer pool is used to continuously extract the corresponding experience from the previous learning species to speed up the reaction speed of learning. In the simulation species, through continuous training and comparison with DQN, NAF and DQN algorithms, it is concluded that this algorithm is better than the other two. The algorithm converges faster, and the later results are more stable, which reflects the saving of the total waiting time of each vehicle in a single intersection environment.

6. References
[1] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529.
[2] Sun R, Tatsumi S, Zhao G. Multiagent reinforcement learning method with an improved ant colony system [J]. SMC, 2001: 1612-1617
[3] Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with Deep Reinforcement Learning[J]. Computer
[4] Silver D, Lever G, Heess N, et al. Deterministic policy gradient algorithms[C]//ICML. 2014.
[5] Dorigo M, Stützle T. Ant colony optimization: overview and recent advances [M]//Handbook of metaheuristics. Springer, Cham, 2019: 311-351
[6] Volodymyr M, Koray K, David S, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015: 529-533.
[7] Peng P, Yuan Q, Wen Y, et al. Multiagent BidirectionallyCoordinated nets for learning to play StarCraft combat games[J]. arXiv preprint arXiv:1703.10069, 2017
[8] Omidshafiei S, Pazis J, Amato C, et al. Deep decentralized multi-task multi-agent reinforcement learning under partial observability [C]//Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017: 2681-2690.
[9] Hitoshi I, Yasuaki K, Shoko M, et al. Swarm reinforcement learning method based on ant colony optimization [J]. IEEE International
[10] Stanley Kenneth O, Miikkulainen Risto. Evolving neural networks through augmenting topologies.[J]. Evolutionary Computation,2002,10(2)
[11] Nardelli N Gabriel S, Zeming L, et al. Value Propagation Networks [J]. ArXiv abs, 2018:11199.
[12] Foerster J, Farquhar G, Afouras T, et al. Counterfactual Multi-Agent Policy Gradients[J]. 2017.