Dueling Double Deep Q-Network for Adaptive Traffic Signal Control with Low Exhaust Emissions in A Single Intersection

Shu Fang1,*, Feng Chen1 and Hongchao Liu2
1University of Science and Technology of China, Hefei 230022, China
2Texas Tech University, Lubbock, TX 79409-1023, USA
*fionarrr@mail.ustc.edu.cn

Abstract. In order to reduce traffic exhaust emissions caused by the large quantities of vehicles, this paper studied the traffic signal control (TSC) model with low exhaust emissions on the basis of the deep reinforcement learning. In this study, the Dueling Double DQN with prioritized replay (DDDQN-PR) algorithm we proposed was combined with the Double DQN, Dueling DQN, and prioritized replay to achieve the goal of low exhaust emissions of TSC. The agent was trained in traffic simulator USTCMTS2.1 in a single intersection. The experimental results show that the performance of DDDQN-PR was significantly better than the other four algorithms, not only in data efficiency but also in final performance.

1. Introduction
Due to the rapid development of Chinese economy and the continuous improvement of urbanization, the number of vehicles has elevated sharply. Traffic congestion has become increasingly serious, and environmental contamination caused by exhaust emissions of vehicles also has attracted more and more attentions from the government and the public. It is no doubt that different vehicle operating modes can result in various vehicle exhaust emissions. Therefore, improving and developing the operation modes is of great importance for vehicles to reduce exhaust emissions.

Traffic signal control (TSC) can effectively reduce traffic congestion and traffic exhaust emissions, as well as avoid the traffic accidents. Thus, improving and optimizing TSC is an effective method to reduce the vehicle exhaust emissions. Fixed time (FT) cannot handle the increasing traffic flow well, therefore implementing and adaptive TSC according to real-time traffic conditions has been employed intensively. Deep Reinforcement Learning (DRL) has been an ideal approach in TSC due to its capability to learn the dynamics of complex problems from interactions with the environment [1]. In order to design and develop the TSC, which is based on the reinforcement learning (RL), to achieve adaptive TSC with low exhaust emissions, the DDDQN algorithm we proposed was combined with the double DQN, Dueling DQN, and prioritized replay to achieve the goal of low exhaust emissions TSC.

2. Deep Reinforcement Learning Foundation
Q-learning is one of reinforcement learning algorithms which was proposed by Watkins and his co-workers [2]. It can be efficiently used to solve the Markov Decision Processes (MDP), which includes...
five components, including the state space $S$, action space $A$, reward $R$, transition probability $P$ and a discount factor $\gamma$.

In the case of large state spaces and action spaces, it is impossible to learn the Q value estimation of each state and action pair independently as standard tabular Q-learning. Therefore, deep neural network was used for DRL to model the components of RL.

2.1. Deep Q-Networks

Deep Q-Networks (DQN) [3] is a multi-layered neural network which is used to approximate action value function. The input of DQN is our state, and the output is the Q value of all actions in that state. DQN addresses these instabilities by using two insights of experience replay and target network [6], which enable relatively stable learning of Q values. Generally, the loss function of DRL is defined as the following equation:

$$L_i(\theta_i) = \frac{1}{2} \mathbb{E}_{(s_t, a_t, r_t, s_{t+1})}[(r + \gamma \max_a Q'(s_{t+1}, a_{t+1}; \theta_i^{-}) - Q(s_t, a_t; \theta_i))^2]$$  \hspace{1cm} (1)$$

Where $\theta_i$ is the main Q-network parameter and $\theta_i^{-}$ is the target Q-network parameter at iteration $i$. It is noted that $\theta_i^{-}$ can be updated with the $\theta_i$ in every $C$ step.

2.2. Double DQN

Conventional Q-learning is affected by an overestimation bias, because of the maximization step, which is harmful to learning. To solve this problem, Double DQN [4] achieves the elimination of overestimation by decoupling the selection of target Q-value actions and the calculation of target Q-value. The update target for Double DQN algorithm is obtained by the next formula:

$$Y_t^{DQN} = r_{t+1} + \gamma Q'(s_{t+1}, \arg \max_a Q(s_{t+1}, a_t; \theta_i^{-}); \theta_i^{-})$$  \hspace{1cm} (2)$$

2.3. Dueling DQN

The dueling network is considered as a neural network architecture proposed by Wang and his co-workers [5]. It divides Q network into two streams of computation, value function $V (s)$, and Advantage Function $A (s, a)$. $V (s)$ is just related to the state $S$, and has no connection with the action $A$. $A (s, a)$ is both related to the state $S$ and the action $A$. By using this approach, Dueling DQN achieves faster training over DQN. The dueling network is defined with the equation:

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + (A(s, a; \theta, \alpha) - \frac{1}{|\alpha|} \sum_{\alpha'} A(s, \alpha'; \theta, \alpha'))$$  \hspace{1cm} (3)$$

Where $\theta$, $\alpha$ and $\beta$ are the convolution layers parameters, the parameters obtained from the $V (s)$ and $A (s, a)$ streams, respectively.

2.4. Prioritized experience replays

When sampling in DQN, all samples in the experience replay pool have the same probability of being sampled. Different samples in the experience replay pool have different effects on back propagation due to the different TD errors. The larger the error of TD, the greater the effect on back propagation. If the larger absolute value of TD error $|\delta(t)|$ samples are easier to be sampled, the algorithm will be easier to converge. Prioritized Replay DQN [6] stores the value of this priority in the experience replay pool, given that the priority of each sample is proportional to $|\delta(t)|$ according to the absolute value of TD error of each sample.
3. Model Design

The single agent simulates a simple four-way intersection, which is illustrated in Fig. 1, which includes a lane for left-turn lane, two straight-going lanes, as well as a lane for right-turn and straight-going in each direction of the intersection. The precondition of applying reinforcement learning into TSC is determined by the appropriate state space S, action space A, and reward function R.

![Fig. 1 The single four-way intersection](image)

3.1. State Space

The description of traffic state should contain enough information, such as the vehicles’ position, speed, acceleration, and current phase information. We defined the state which consists of a binary matrix for vehicle positions (1 and 0 indicate the vehicle in this cell and not in this cell, respectively) and the current phase representation (1, 0 and 0.5 indicate the permitted, banned and yellow light, respectively).

The position state matrix of the single intersection is illustrated in Fig. 2a. Similarly, it’s easy to get the same size normalized velocity matrix corresponding to the position matrix. The state matrix obtained by a simulation step, which can be expressed as $\mathbf{s} = [\mathbf{P}, \mathbf{V}]$. In this experiment, the state matrix $\mathbf{s}_t = [s_{t-3}, s_{t-2}, s_{t-1}, s_t]$ obtained from previous four simulation steps which is taken as the current traffic state at the time step $t$. $\mathbf{s}_t$ includes static information and dynamic information, which can depict the traffic state more accurately and profoundly.

![Fig. 2 a Position state matrix](image) ![Fig. 2 b Velocity state matrix](image)

3.2. Action Space

The action in this experiment represents whether the current traffic signal phase changes. The action set is $A = \{0, 1\}$, where $A=0$ means to keep the phase unchanged with the period of $\tau$ (set 4 simulation steps); and $A=1$ means to jump to the next phase.

3.3. Reward Definition

Let $\mathbf{e}_{i,t}$ be the $i$th vehicle’s the $i$th emissions at the time step $t$, and $\mathbf{E}_t$ indicates the total cumulative emissions for all the vehicles and three kinds of vehicle exhaust emissions (the weighted sum of $NO_x$, $HC$ and $CO$) in the road network at the time step $t$. In USTCMTS2.1 we built up the MOVES vehicle emission model.
The intention of the reward function was used to reward the agent positively, where the emissions decrease between time step $t$ and $t + 1$. Thus, the agent is supposed to keep low emissions to receive higher scores. Subsequently, this reward function accomplishes the goal of reducing the vehicle’s emissions at an intersection.

$$E_t = \sum_i \sum_j w_{ij} e_{ij,t}$$  \hspace{1cm} (4)

$$r_t = E_{t-1} - E_t$$  \hspace{1cm} (5)

### 3.4. Network Architecture

The network architecture of DDDQN-PR shown in Fig. 3. The position and speed matrix were combined as two input channels, and we utilize three convolutional layers to extract useful information, then connect a dueling network to get the action.

![Fig. 3 DDDQN-PR Network Architecture](image)

### 4. Simulation and Analysis

In this section, we present an evaluation of DDDQN-PR as a proposed solution for a single agent in low exhaust emission TSC using the traffic simulator USTCMTS2.1, which was developed by the laboratory of the data fusion & intelligent traffic system, University of Science and Technology of China. USTCMTS2.1 is a microscopic and continuous traffic simulation software which was designed to deal with large road networks. The network shown in Fig.3 is implemented using Python and TensorFlow libraries. USTCMTS2.1 uses C# as a programming language, while C# and Python communicate uses the IronPython library.

#### 4.1. Simulation Settings

Set the phase as a typical four phases control scheme (Fig.4). Setting a cell length to be 9 m and each road to be 288 m. The vehicle length is 5 m and the minimum gap between vehicles is 2.5 m. The road minimum speed is 13.89 m/s and the maximum vehicle speed is 22.22 m/s. The range of green time is 8 to 60 seconds. The traffic flow in (N-S, S-N, E-W, W-E) is 665 (veh/h), 704 (veh/h), 638 (veh/h) and 693 (veh/h), respectively.

![Fig. 4 The typical four phases](image)
4.2. Algorithm hyper-parameters
Table 1 lists the hyper-parameters used. Each episode represents 3600 simulation seconds of traffic simulation and the performance is evaluated by the average exhaust emission that vehicles travel through the intersection, which is the sum of all $E_t$ divided by the number of vehicles in the episode.

| Parameter                  | Value |
|----------------------------|-------|
| Episodes                   | 1000  |
| Total steps                | 3600000 |
| Replay Memory size         | 30000 |
| Batch size                 | 128   |
| Learning rate              | 0.00025 |
| Target network update      | 10    |
| Train frequency            | 128   |
| Discount factor $\gamma$  | 0.99  |
| Exploration $\epsilon$    | 1.0 → 0.01 |
| Prioritization importance sampling $\beta$ | 0.4 → 1.0 |

4.3. Simulation Results
We implement and evaluate the combinations of DQN, Double DQN, Dueling DQN, and prioritized replay, to assess impact of each component on the performance:
- DQN with random experience replay.
- Double DQN with random experience replay.
- Dueling DQN with random experience replay.
- DDDQN-RR Double Dueling DQN with random experience replay.
- DDDQN-PR Double Dueling DQN with prioritized experience replay.

Fig.5 presents the results of our experiments. It is clear that the DQN, Double DQN, Dueling DQN, DDDQN-RR, and DDDQN-PR were used to achieve a better performance than FT baseline. The performance of DDDQN-PR is significantly better than any of FT, both in data efficiency, as well as in final performance. In the final evaluations of the agent, after the end of training, DDDQN-PR achieves a median value of 0.296; in the FT we measured a value of 0.552. In Table 2 we compare these values to the median values of the individual baselines.

![Figure 5: Average Emission in training progress](image)
Since DDDQN-PR integrates several different ideas into a single agent, we need to compare different combinatorial experiments to understand the contribution of the various components, in the context of this specific combination. Prioritized replay was the most crucial components of DDDQN-PR, in that removing the component caused a large drop in median performance. DDDQN-PR explores a little more efficiently than DDDQN-RR as it benefits from the bootstrapped prioritized updates, which utilizes experiences with priority to update its inner parameters at each learning step. In the case of Double DQN and Dueling DQN, the median performance and the convergence of the components are similar. Removing either of the components caused a drop in median performance.

Table 2. Median Average Emission of the agent snapshots for DDDQN-PR and baselines

| Agent         | Median Average Emission (g) |
|---------------|-----------------------------|
| Fix-Time      | 0.552                       |
| DQN           | 0.391                       |
| Double DQN    | 0.371                       |
| Dueling DQN   | 0.373                       |
| DDDQN-RR      | 0.348                       |
| DDDQN-PR      | 0.296                       |

5. Conclusion

In this paper, DDDQN-PR was used to solve the TSC problem with low traffic exhaust emission. Experimental results illustrate that the DDDQN-PR algorithm improves the performance of DQN algorithm, which is also superior to FT and another single-component Q-learning algorithms. In this paper the experiment is just set up for a single intersection, further experiments need to be done in real traffic network with multi agents.

Acknowledgments

This work was funded by The Key Research & Development Project of Anhui Province (1804b06020376) and The National Key Research & Development Program of China (2017YFC0840206) respectively.

References

[1] Genders W, Razavi S. Using a Deep Reinforcement Learning Agent for Traffic Signal Control [J]. 2016.
[2] Watkins C J C H, Dayan P. Q-learning. Machine learning, 1992. 8 (3-4): 279-292.
[3] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518 (7540): 529.
[4] Van Hasselt H, Guez A, Silver D. Deep Reinforcement Learning with Double Q-Learning [C]// AAAI. 2016, 2: 5.
[5] Wang, Z., de Freitas, N., Lanctot, M.: Dueling Network Architectures for Deep Reinforcement Learning. arXiv (9), 1(16 (2016)
[6] Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. CoRRabs/1511.05952 (2015), http://arxiv.org/abs/1511.05952.