Cooperative Deep Reinforcement Learning for Traffic Signal Control

Mengqi LIU
Beijing University of Posts and Telecommunications
Beijing, China
victorialiu0628@gmail.com

Jiachuan DENG
Purdue University
Indiana, US
deng159@purdue.edu

Ming XU∗
Tsinghua University
Beijing, China
mxu@tsinghua.edu.cn

Xianbo ZHANG
Northeastern University
Shenyang, China
neu_zhang_xianbo@hotmail.com

Wei WANG
Northeastern University
Shenyang, China
neu_wangwei@hotmail.com

ABSTRACT
Traffic signal control plays a crucial role in intelligent transportation system. However, the existing approaches for traffic signal control based on reinforcement learning mainly focus on traffic signal optimization for single intersection. We propose a deep-reinforcement-learning-based approach to collaborative control traffic signal phases of multiple intersections. For the state representation, we use comprehensive road network information as input, and for each controller (intersection), sufficient autonomy is given so as to enable it to adapt to different kinds of intersections. Besides, we design a reward considering driver patience and cumulative delay, which can better reflect the reward from the road network. Based on the above opinions, a variant of independent deep Q-learning is introduced, so that multiple agents can be applied experience replay to speed up the training process.

KEYWORDS
Traffic Signal Control, Deep Reinforcement Learning, Independent Q-Learning, Simulation

1 INTRODUCTION
People’s living standards are increasing, which leads to the increasing of the demands of private cars. In this regard, in order to alleviate the increasing traffic pressure, we should strengthen the urban traffic signal management. The reasonable traffic lights set, not only conducive to increasing traffic flow, reducing traffic safety hazards and travel time, but also reducing traffic energy consumption, urban air pollution and peoples’ travel costs. There are three possible solutions for this problem.

∗This is the corresponding author.

(1) Macro-control, i.e. national policy, to limit the number of vehicles on the road network. For example, even and odd-numbered license plate. But the promulgation of the policy involves too many aspects, and not within our capabilities.

(2) From the perspective of road infrastructure, we can build viaducts, underground fast-ways, etc., to increase the road network capacity. But this method will cost too much, and in the early time, it will reduce the road network capacity.

(3) On the basis of the existing infrastructure, we can improve the operational efficiency of the road network and our ability to manage the road network.

We are more inclined to the third solution, and we decide to make some improvements from the point of view of traffic signal controlling.

The reinforcement learning technique has already been widely applied to the problem of traffic signal control [4, 9, 20]. However, the scope of research on the problem of traffic signal control is still limited to the behavior of single intersection, and that each intersection is independent of each other. In order to take the potential relationship among different intersections into consideration and not go too far, intersections need to be managed in one certain region at one time, and for each intersection inside this region, they need to be able to perceive strategies of others. To illustrate the relationship among intersections, consider the following examples.
As shown in Figure 1, when traffic flow at intersection A goes high, intersection B and C may need to increase the duration of West-East Green Light and North-South Green Light, respectively, so that there won’t be future congestion at intersection B and C. Besides, since we are considering the problem of traffic signal control which can be simplified to manage vehicles on the road. Intersections that are far away from each other, i.e. out of a certain region, will not be taken into consideration, this because vehicles can’t move to distant intersections without passing through other intersections between them. To summarize, the impact between different intersections will decrease as distance goes far, distant intersections contribute too little to be counted. By limiting the scope of management in a certain region, we can achieve high accuracy as well as high efficiency at the same time. A powerful approach to tackle the challenge of perceiving strategies is Independent Q Learning (IQL) [24], in which the agent is trained one by one, and delivers the training results to other agents, so that these cooperative agents can sense each other’s policies and can handle the collaborative task. However, IQL doesn’t perform well when combined with Experience Replay [16] which will be explained in Section 2.1.

Except for the above challenges, there are still some other challenges, as shown below, occur when mapping the problem to the language of reinforcement learning. To tackle all these challenges, we propose the Cooperative Deep Reinforcement Learning (CDRL).

- State Representation. Previous researches tend to represent the network with some attributes such as the queued length, average traffic flow, etc. Although these attributes are very abstract expressions for road network information, and are widely recognized in the field of traditional transportation system, they can’t be used as inputs to the Deep Neural Network (DNN). Because the inputs of DNN are designed to be comprehensive, unmodified information.
- Action Space. The traditional operation model of traffic signal phase is not unified, for example, some intersections use through phases before advance left phases, while some others are just the opposite.
- Rewards. There are some common reward definitions in these fields, change in the number of queued vehicles, change in delay, etc.. However, none of them can reflect the psychological changes in the process of traffic congestion. For example, if we are waiting for a regular red light of about 40 seconds, we are generally not impatient. In the contrast, if we are caught in traffic jam and have been blocked for 10 minutes, most of us will feel very unhappy.
- Computing Complexity. Many problems in machine learning may be influenced by the curse of dimensions, as the dimension of the data increases, the computing complexity increases exponentially.

For each of these difficulties, the contributions of CDRL are as follows:

- The action space of CDRL is fixed, however, the order of actions are generated by CDRL without restriction. For each intersection, our action space definition is supposed to give it enough autonomy, so that it can learn the potential optimal policy despite the changing environment.
- We propose a reward function to describe the exponential relationship between the patience of the driver and the time waiting. So as to better describe the road traffic from the perspective of human beings.
- CDRL use multiple agents, i.e. one agent only represent one intersection, so that it can prevent the state space of a single agent from being too large and growing exponentially as the number of intersection increases. Besides, we use convolution-based residual network to speed up the training process while preventing the possibility that the training results might be affected as the neural network goes deeper.

2 BACKGROUND

2.1 Independent Q-Learning with Experience Replay

2.1.1 Q-Learning [28]. Q-learning is aimed to calculate the utility values when executing an certain action under the given environment by iteratively updating the Q-values with Equation (1).

\[
Q(s, a) = Q(s, a) + \alpha (r + \gamma \max_{a'} Q(s', a') - Q(s, a))
\]  

In Equation (1), \(Q(s, a)\) represents the utility value of taking action \(a\) under state \(s\). Learning rate \(\alpha\) determines how new estimates are weighted against the old ones, and \(\gamma\) denotes the discount factor, which controls the degree between future rewards and current rewards.

2.1.2 Experience Replay. At each training step, the agent’s experience tuple \(e_t = (s_t, a_t, r_t, s_{t+1})\), where \(r_t\) is the reward (utility value) obtained by taking action \(a_t\) under state \(s_t\) and \(s_{t+1}\) is the next state, resulted by action \(a_t\) and current state \(s_t\). These experience tuples are stored in order, older before newer, and when the replay memory goes full, it will automatically pop the oldest tuple. Training tuples will be randomly selected from the replay memory for Q-value updating, so as to stabilize the training process, enhance sample efficiency and prevent correlation between samples.

2.1.3 Independent Q-Learning. In IQL, each agent learns its own policy independently and treats other agents as part of the environment, and each agent is able to track the policies of other agents in real time. From the perspective of single agent, the environment is unstable. This is because the other agents in the same system are also changing (updating). It will be difficult for the system to achieve convergence. Despite the non-stationarity issues, IQL can learn well as long as each agent can track other agents’ policies in real time.
2.2 Deep Reinforcement Learning (DRL)

Q-Learning can be extended to DRL with Equation (2), where θ are the weights of DNN and \( Q(s, a|θ) \) is the neural network approximation of \( Q(s, a) \) in Equation (1). To be specific, the weights are updated by minimizing Equation (2), which is also called loss function.

\[
L(s, a|θ) = (r + γ \max_a Q(s', a|θ) - Q(s, a|θ))^2
\]  

(2)

2.3 Deep Residual Network (ResNet) [12]

ResNet is modeled to speed up training process while providing a reasonable training results, regardless of the depth of neural network. This process is achieved by performing the following computation, Equation (3).

\[
x_{t+1} = x_t + F(x_t, W_t)
\]  

(3)

Here \( x_t \) and \( x_{t+1} \) are the input and output of the \( l \)-th Residual Unit, respectively. \( W_t \) is the corresponding weights (and biases). \( F \) represents the residual function, e.g. the stacked two convolutional layers in [12].

3 METHOD

3.1 State Space

For the convenience of display, we give another simplified road network as Figure 2 with two intersections, instead of the original four intersections in Figure 1. For the state representation, CDRL will transfer the state of the entire road network into four image-like representation: vehicle position, vehicle speed, traffic signal phases (TSP) of other intersections and TSP of the current intersection.

3.1.1 Vehicle Position. Following the previous work [11, 26], we use Boolean values to mark the presence of the vehicles, 0 for not exist, 1 for exist, as shown in Figure 3(a).

3.1.2 Vehicle Speed. The speed of each vehicle is normalized by the speed limitation, which leads to the real numbers represented in Figure 3(b).

3.1.3 TSP of Other intersections. Figure 3(c) represents the TSP of other intersections. When the number of intersections increases, the area of TSP in the third image, Figure 3(c), should also increase, correspondingly. For example, as shown in Figure 1, if the current intersection is \( A \), then the other intersections should be \( B \), \( C \) and \( D \), which means all of these three TSP should be shown in the third image.

3.1.4 TSP of the Current intersection. As shown in Figure 3(d), it represents the TSP of intersection \( A \). Besides, the fourth image should always represents the TSP of one intersection, the current intersection, no matter how many intersections there are in the current road network. In the case of example in Section 3.1.3, the current intersection is \( A \), and there should only be the TSP of intersection \( A \) appeared in the fourth image.

Because the traffic signal lights don’t occupy the driving spaces, nor do they occupy the non-street spaces. In order to avoid interference, we can’t mark the TSP at the entrance / exit of each lane, or the center of the intersections. In our work, the TSP are represented by filling the corresponding lanes as 4 for red light and 2 for green light. In this example, we take intersection \( X \) as the current intersection, and intersection \( Y \) as the other intersection.

3.2 Action Space

Once the state is observed, agent has to choose one appropriate action based on the definition of action space. Formally, the set of all possible compass directions for each agents is defined as \( A = \{\text{NSG, EWG, NSLG, EWLG}\} \), where NSG stands for North-South Green, EWG stands for East-West Green, NSLG stands for North-South Left Green, and EWLG stands for East-West Left Green. For each time step, the agent can only choose one of these actions, and the other three directions will be set as Red by default. In this case, it will be possible for the agent to determine whether it should continue the previous action or change to another one based on the current state, so as to give the agent more autonomy. Thus, agents can make more flexible decisions under changing states.
For example, at time step $t$, $a_t = \text{NSG}$, assume the north-south direction is slightly congested, and there is no vehicle in the east-west direction; and at time step $t + 1$, assume the chosen action will be $a_{t+1} = \text{NSG}$, the north-south direction continues to jam, and there is one single vehicle in the east-west direction; then at time step $t + 2$, the agent with a high degree of autonomy will continue NSG, however, agent with strict pre-defined rules may choose EWG under certain restrictions, such as the duration of red light shouldn’t exceed 40 seconds.

Additional yellow phase is added before the changing phase for safety assurance. To be specific, yellow phase will only be added before the phase whose upcoming phase differs from its current phase.

### 3.3 Reward

The reward in reinforcement learning is used to evaluate the impact of an action on its environment. The objective of learning is to maximize the cumulative reward that agents receive in the long run. In order to achieve global optimum, a common approach is to define the reward as the average delay, given by

$$r_t = \frac{1}{N} \sum_{i=1}^{N} t^i$$

$r_t$ in Equation (4) represents the reward at each time-step, and $t^i$ refers to the current waiting time for each driver, while $N$ is the total number of drivers.

However, when such rewards are used as learning objectives, the interests of a small number of vehicles may be overly damaged. Specifically, in order to minimize global vehicle delays, DRL may tend to keep some road segments with heavy traffic loads being unobstructed, which may cause the extension of red signal phases on some other segments with small traffic flows. Unfortunately, such extension usually last long. In order to avoid similar situations, the reward function, inspired by the well-known U.S. Bureau of Public Roads (BPR) function in transportation planning, is defined as Equation (5),

$$r_t = \frac{1}{N} \sum_{i=1}^{N} d[\eta - \eta(t^i)^\tau]$$

where $r_t$ is the reward at time-step $t$, $t^i$ refers to the current waiting time for driver $i$, $d$ represents one unit of time, $C$ denotes the generally acceptable waiting time, $\eta$ and $\tau$ are constants, which are normally set as $\eta = 0.15$, $\tau = 2$ according to BPR.

### 3.4 CDRL with Experience Replay

#### 3.4.1 Cooperation among Agents

In IQL, although each agent will treat others as part of the environment, non-stationary is brought by the impact among multiple agents. In particular, since multiple agents are interacting with the environment, the combination of any single agent and the environment can no more be thought as stable. In this case, if we still want to let multiple agents interact with the environment all together, all of the agents involved need to be taken into consideration, and will certainly cause dimension curse. In CDRL, agent will still learn its own policy and treat others as part of the environment. However, to prevent non-stationary, there will only be one agent training at each training episode, the others will behave based on their pre-viously generated policies. In the training process, agent doesn’t need to communication with the other agents. After each training
episode, the final policy will be delivered to the other agents, while the learning process itself will not be shared.  

3.4.2 Combined with Experience Replay. Each agent has its own replay memory. Since tuples \( e_t = (s_t, a_t, r_t, s_{t+1}) \) generated by different agents are different, especially on their state representation \( s_t \) and \( s_{t+1} \). If we simply apply one replay memory to all agents, the tuple that is randomly generated from the reply memory is unlikely to reflect the current agent.

Figure 4 shows the architecture of the CDRL. Four 10x10 image-like matrix will be combined as a 10x10x4 dimension tensor as the input for each agent. Specifically, assuming that we are currently training agent \( A^2 \) (see Figure 1). Each of the four agent will get their unique 10x10x4 dimension tensor as input. However, only agent \( A \) will get the reward, which means only the ResNet and Replay Memory of \( A \) will be updated. After a certain iterations, the latest policy of \( A \) will be distributed to the other three agents, and the training intersection will also change to another intersection.

In this way, CDRL avoids the non-stationary problem caused by changing environment, thus can be further combined with experience replay without causing troubles.

3.5 Algorithm and Optimization
Algorithm 1 shows the training process of CDRL. We first initialize all intersections then we get actions from all intersections, what’s more, there will be one intersection being trained and updated for each training period.

4 EXPERIMENTS
4.1 Setting
To simulate a road network, traffic micro-simulator SUMO v0.22 [13] is used in the experiments. It provides a Python API to interact with the program of neural network. We implemented CDRL model in TensorFlow Python framework.

The simulated traffic network is built as a network with the shape of “#”, which consists of four intersections. Each intersection connects to four road segment, and each road segment includes three outgoing lanes in two opposite directions: left turning lane, straight lane and right turning lane, as is shown in Figure 1. For simplicity and clarity, four lanes in different directions are shown specifically, and other parts of the traffic network are arranged in the same way. During the simulation process, each road segment can be viewed as an origin or a destination, and an certain amount of traffic is generated according to the pre-defined Origin-Destination matrix.

To evaluate the effectiveness of CDRL, we compare CDRL with three baseline algorithms:

1. Self-Organizing Traffic Lights (SOTL) [7]: a SOTL controller turns the state of signal lights according to the elapsed time and the number of vehicles in queues.
2. Q-Learning [3]: a tabular reinforcement learning method widely used in signal control, its state and action spaces

\[ Algorithm 1: \text{CDRL Training Algorithm} \]

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replay memory ( D ) with size ( M )</td>
<td>Actions for all intersections ( \text{trainIter} = 0 )</td>
</tr>
</tbody>
</table>

// initialize all intersections
Initialize replay memory \( D \) with size \( M \)
Initialize action-value function \( Q \) with random weights
// get actions
if \( \text{probability} < \epsilon \) then
    select action \( a_t = \max_a Q(s_t, a_t; \theta) \)
else
    select random action \( a_t \)
end
Execute action \( a_t \) in simulator and observe reward \( r_t \) and next state \( s_{t+1} \)
// train and update
get actions
put all training instances \((s_t, a_t, r_t, s_{t+1})\) into \( D \)
if \( \text{terminal} s_{t+1} \) then
    set \( y_j = r_j \)
else
    set \( y_j = r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) \)
end
Perform a gradient step on \((y_j - Q(s_j, a_j; Q_t))^2\) with respect to \( \theta \)
// main loop
initialize all intersections
repeat
    for iteration times of each intersection \( t (1 \leq t \leq p) \) do
        get actions from all intersections \( I \)
        train and update intersection \( I_{\text{trainIter}+1} \)
    end
    \( \text{trainIter} + 1 \)
until \( \text{trainIter} > n \);

are small enough, and the policy function is represented as a table.

3. Deep Q Network (DQN) [19]: each junction is controlled by one agent, and each agent determines the alternation of TSP only basing on traffic situation of its direct linking roads.

Note that, both SOTL and Q-Learning need to take the traffic features as input, such as the length of queue, the traffic flow or average speed of lanes, whereas, DQN solely needs original traffic information as in CDRL, see Section 3.1. In the training process, agents of reinforcement learning and DQN are trained simultaneously and independently and do not distribute policies with other agents. An episode is a simulation process that lasts 30 minutes.

4.2 Results
Figure 5 presents the variation of rewards with the increase in the number of episodes during training process. As is shown, the
SOTL still keep unimproved, for it is a pre-defined method. In Q-learning, DQN and CDRL, the reward grows at different rates and then stabilizes. In contrast, the CDRL converges slowly, however, when the convergence is complete, the CDRL behaves slightly better than the other two.

Figure 6 shows a change in the travel delay with the increase in the number of episodes. In addition to SOTL, its overall performance shows a declining trend. We can observe that the convergence rate of CDRL is the slowest in Q-Learning, DQN and CDRL. However, once convergence, the CDRL achieves the minimum travel delay. By comparing Figure 5 and Figure 6, it is not difficult to find that the CDRL achieves a significant improvement in performance by sharing policies with others in each turn. By comparing Q-Learning and DQN, we can see that DQN converges more slowly, but it performs better after convergence. In conjunction with the performance of CDRL, it is not difficult to find that deep neural networks are more capable of capturing traffic flow and improving performance.

$1\text{ turn} = 1500\text{ episodes}$

Figure 7 shows the relationship between the overall cumulative delay and the running time in the 7500-th episode. CDRL performs best. This further confirms the importance of multi-agent cooperation for good performance.

Next, in order to clearly understand the impact of different rewards. We compare the proposed reward to the baseline reward, average delay, based on the state space and action space defined in CDRL (see Section 3.1, Section 3.2). Specifically, we measure these two rewards by the maximum delay, which represents the delay time of the vehicle with the greatest delay in the road network. As shown in Figure 8, these two rewards perform similar in the early terms of training, and as the running time increases, the proposed reward significantly reduces the maximum delay, however, the baseline reward shows the trend that contributes to the maximum delay, which means that some vehicles have been blocked for quite a long time.

Based on the above experiments, we can see that by applying the proposed reward, the CDRL can usually minimize the average delay of the vehicle as well as reduce the likelihood that a small number of vehicles will be blocked for a long time.
5 RELATED WORK

5.1 Traffic Signal Control

Early research in traffic signal control was largely limited by the performance of the traffic simulator [2, 5, 25, 29]. From the beginning of 2000, traffic simulation ushered in a rapid progress, simulation tools were developed to be more and more complex and realistic, which can almost reproduce the real world traffic behavior. However, the emergence of reinforcement learning has broken the traditional solution and demonstrated remarkable results [10, 17]. The use of reinforcement learning to solve traffic signal control problems is mainly distinguished by differences in state space, action space, rewards, network topology and simulators. 4 The definition of state space is often based on traffic attributes, such as the queued length [1, 2, 6, 29], average traffic flow [3, 4], cumulative delay [11, 21] and so on. The action space is often defined by the signal phase of all signal phases or green lights [1, 4, 6]. And the reward is often defined by changes in average delay [3] and changes in the queued length [1, 4, 6].

5.2 Deep Reinforcement Learning

Reinforcement learning is aimed to maximize the long-term rewards by performing a state-action policy. However, when the state space goes too large to handle, function approximators, neural networks, can be used to approximate value functions. To summarize, Deep learning is responsible for representing the state of the Markov Decision Process, while reinforcement learning is supposed to take control of the direction of learning.

Regarding previous research, the road network state space is defined based on the abstract information of the existing traffic attributes, which may be lack of useful information for defining the state space. For example, the definition based on the queued length ignores the speed of the vehicle, since both of the following are likely to have a larger queued length, a large number of vehicles passing smoothly and quickly through the intersection, and a large number of vehicles without movements, i.e. traffic congestion.

In order to comprehensively extract road network information, it is necessary to use DRL, not only because the restrictions of reinforcement learning when dealing with large state space, but also because of the advantages of convolution neural network (CNN), it requires little pre-processing for the input, and can further develop its own features. In addition, convolution neural networks have shown charming results in the area of computer vision [14]. Since the road network itself can sometimes be represented by images, it will be quite convenient to apply DRL to our problem. DRL itself has already successfully applied to the problem of managing the strategic dialogue [8] and the problem of control actions in progress [15]. There are also more complex domains like Go games [22]. In addition, researchers have studied various ways to improve DRL performance such as double Q-learning [27] and asynchronous learning [18]. However, it is still very difficult to apply the DRL to multi-agent cooperation, and the best existing results exist only under the cooperation of two DQNs [23].

Table 1: Hyperparameters for CDRL

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episode</td>
<td>450 iteration</td>
</tr>
<tr>
<td>Max train iterations</td>
<td>7500 episode</td>
</tr>
<tr>
<td>Minibatch size</td>
<td>32</td>
</tr>
<tr>
<td>Replay Size</td>
<td>3000</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.0001</td>
</tr>
<tr>
<td>Update rule</td>
<td>ADAM</td>
</tr>
<tr>
<td>Initial $\epsilon$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\epsilon$ decay</td>
<td>5e-5</td>
</tr>
<tr>
<td>Target network update</td>
<td>150</td>
</tr>
<tr>
<td>Agent network update</td>
<td>1500 episode</td>
</tr>
</tbody>
</table>

Finally, all previous studies have used computer simulations because real-world experiments are not feasible for a variety of reasons.

6 CONCLUSION

In this paper, we proposed a traffic signal control method for multiple junctions basing on cooperative deep reinforcement learning. At the core of the proposed method is a multi-agent framework, in which each agent utilizes a convolution-based residual network to solve the control optimization problem with continuous state space in a cooperative manner. In addition, we introduced a more realistic reward function from a perspective of Traffic Psychology. Simulation experiments show that the proposed method outperforms other baseline traffic signal control methods.

7 FUTURE WORK

In the future work, we will extend deep multi-agent reinforcement learning framework to collaborate more agents and apply to traffic signal scheduling on a larger scale of road network. In order to prove robustness, we will also apply our framework to the road network with different topologist and different traffic generation model.

A HYPERPARAMETERS FOR CDRL

The hyperparameters for training our CDRL is shown as table1. The max train iterations refers to the total training times, which is $p*n$ in Algorithm1. Minibatch size refers to the number of training samples for each updates. Replay size is the size $D$ of replay memory $M$. Learning rate refers to $\alpha$ as explained in Equation (1). Our update rule for the optimizer is ADAM. Initial $\epsilon$ refers to the initial value of $\epsilon$ for $\epsilon$-greedy exploration, and the decay rate of the exploration is shown as $\epsilon$ decay. Target network update represents the frequency of updating the target Q network in the experiment. And agent network update refers to training period of each intersection as shown in Algorithm1.

REFERENCES


4traffic generation model is involved in simulators


