Journal of Advances in Mathematics and Computer Science

27(3): 1-11, 2018; Article no.JAMCS.41281
ISSN: 2456-9968
(Past name: British Journal of Mathematics & Computer Science, Past ISSN: 2231-0851)

Deep Recurrent Q-learning Method for Area Traffic Coordination Control

Saijiang Shi[1*] and Feng Chen[1]

[1]Department of Automation, University of Science and Technology of China, Hefei 230027, China.

Authors' contributions

This work was carried out in collaboration by both authors. Author SS designed the study, implemented the algorithm and wrote the manuscript. Author FC optimized the method and guided the experiments. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/JAMCS/2018/41281
Editor(s):
(1) Jacek Dziok, Professor, Institute of Mathematics, University of Rzeszow, Poland.
Reviewers:
(1) Grzegorz Sierpiński, Silesian University of Technology, Poland.
(2) Hui Yang, Beijing University of Posts and Telecommunications, China.
(3) Phung Duy Quang, Foreign Trade University, Vietnam.
(4) Oluleye Babatunde, Charleston Southern University, USA.
Complete Peer review History: http://www.sciencedomain.org/review-history/24643

Original Research Article

Received: 26th February 2018
Accepted: 4th May 2018
Published: 16th May 2018

Abstract

In order to improve the performance of Deep Q-learning when dealing with the area traffic control which is a partially observable Markov decision process. This paper introduces Deep Recurrent Q-learning by changing the fully connected network layers to LSTM layers. On the other hand, we use transfer learning to achieve the coordination of multiple intersections in the area. By the simulation experiments, this paper compares the average delay of our algorithm with the Deep Q-learning algorithm for three different saturation flows, respectively. We also compare our algorithm with another two popular traffic signal control algorithms, i.e., Q-learning and fixed time control algorithm. The experiment results show that the performance of our improved Deep Recurrent Q-learning algorithm is better than the other three algorithms.

Keywords: Area traffic coordination control; deep recurrent network; deep Q-learning; deep recurrent Q-learning.

**Corresponding author: E-mail: shisj@mail.ustc.edu.cn;*

1 Introduction

1.1 Background and significance

The problem of traffic congestion has been more and more serious nowadays, causing economic losses and environmental pollution. There are mainly two techniques to solve this problem. One is the construction of infrastructure, but the cost of this method is relatively expensive, and it cannot solve the problem fundamentally. The other is developing an intelligent transportation system to effectively control the urban traffic signals without increasing extra infrastructure. The latter can save costs and alleviate urban traffic congestion effectively. Surveys and statistics show that most of the traffic jams occurred at intersections [1], so it is particularly critical to adopt more intelligent control strategies for urban intersections. This article coordinated the control of the area which is composed of multiple intersections and improved the signal control performance of the intersections.

1.2 Related works and paper outline

The use of Deep Q-learning(DQN) method for optimal control of intersections has achieved good performance. DQN does not need to extract human-crafted features and can solve the problem of dimensional explosion [2]. Richter S et al. proved that the control of the intersection is a Partially Observable Markov Decision Process (POMDP) [3], thus DQN's performance deteriorates due to inaccurate observation status. Some efforts have been made in the state representation to reduce the impact of POMDP. The literature [4] and [5] proposed a method to represent Q-values with deep convolution neural networks. The location and speed of vehicles at the intersection are represented by matrixs. In addition to using the location and speed matrixs, Van der Pol E also added the acceleration matrix of vehicles as the state of intersection [6]. The work [7] directly input the first four frames of the video obtained from intersection and use the deep neural network to formulate these high-dimensional state.

Area traffic coordination control, which is a multi-agent Q-learning(MAQL) problem, can be divided into independent MAQL, partially state-cooperation MAQL, and joint-action MAQL [8]. The independent MAQL treats each intersection in the road network as a single agent. This method does not consider the relationship between the intersections. The partially state-cooperation MAQL achieves mutual cooperation through sharing states between the intersections. But this method ignores the influence of intersections' actions since the active agent i takes influences j's optimal action. We adopt joint-action MAQL because it considers the joint action of the agents. El-Tantawy S et al. presented the development and evaluation of a novel system of multiagent reinforcement learning for an integrated network of adaptive traffic signal controllers (MARLIN-ATSC) [9]. Zhu F et al. proposed the Junction Tree Algorithm (JTA) based reinforcement learning to obtain an exact inference of the best joint actions for all the coordinated intersections [10]. Abdoos M et al. used hierarchical structures to decompose the network into multiple sub-networks and provide a mechanism for distributed control of the traffic signals [11].

This paper introduces Deep Recurrent Q-learning(DRQN) to further reduce the impact of the POMDP. For area traffic coordination control, transfer learning is used to reduce the training time for DRQN.

The rest of this paper is organized as follows. In Section 2, we define reinforcement learning components for a single intersection, and we also introduce the structure of DRQN. In Section 3, we apply DRQN to multi-agents, using transfer learning and max-plus for area traffic coordination control. In Section 4, we evaluate our algorithm by simulations and compare its performance with some popular traffic signal control algorithms. In Section 5, we conclude the whole paper.

2 Q-Learning for Single Intersection

Q-learning is described by the following process: during each iteration, the agent observes its current environment, from which it infers the environment's state S , then executes an action selected from the

action space A . Then the agent transfers to the next state and gets the reward R . The Q-values from S to A is denoted by $Q(S,A)$.

Assume that the state of the agent at time t is s_t ,the action taken by the agent is a_t , the state transitions to the next t+1 moment is s_{t+1} and the agent gets the reward r_t . After that, the agent updates $Q(S,A)$ according to all records (s_t,a_t,r_t,s_{t+1}) to find the best strategy. In this section, we formulate traffic signal control problem as a Q-learning problem.

2.1 State space

Consider the four-phase single intersection shown in Fig. 1. Taking the position, velocity and acceleration matrix of the vehicles in the current intersection as input.

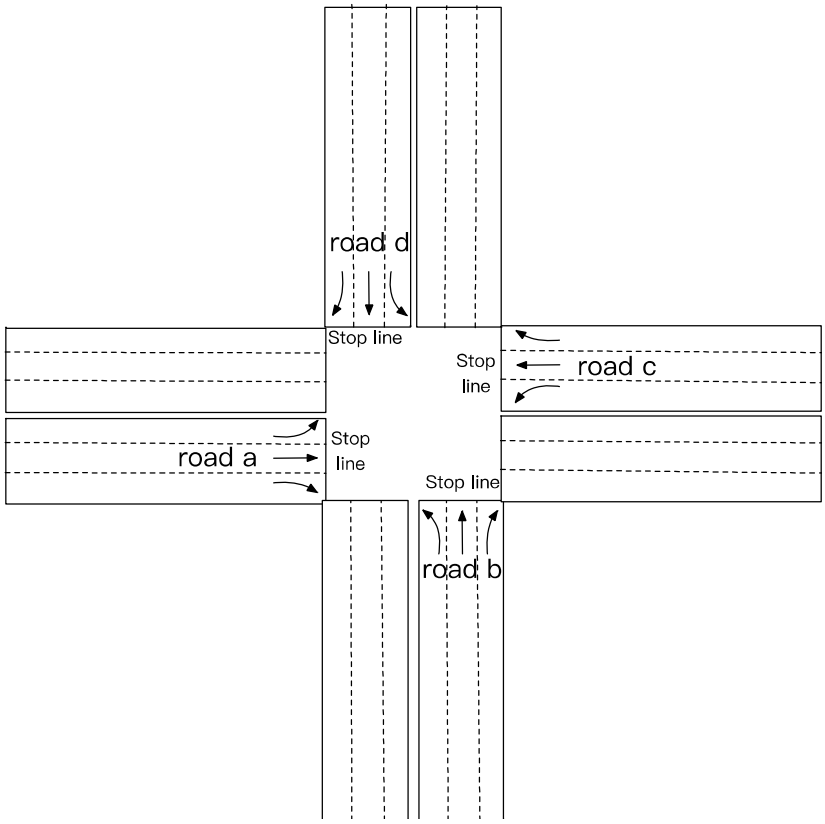

Fig. 1. Four-phase intersection schematic

Take road a in Fig. 1 as an example, suppose the current intersection vehicle is shown in Fig. 2. Consider a section of distance L from the stop line, divide the road into multiple small sections called Cell, and the length of each Cell is C. Then, the position matrix of the current intersection is shown in Table 1. When there is a vehicle in the divided Cell, the corresponding position of the matrix is 1, and vice versa is 0. When a vehicle crosses two Cells, the Cell that has a large proportion of occupancy is selected. The velocity matrix and the acceleration matrix are the same as the position matrix, the matrix values are normalized by the upper limits of velocity and acceleration in the simulation software.

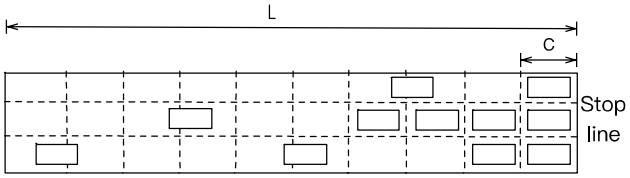

Fig. 2. Vehicle location of road a

Table 1. Position matrix of entrance road a

0	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	1	1	1	1
1	0	0	0	0	1	0	0	1	1

2.2 Action space

In this research, the possible actions of the agent are the traffic signal phase configurations. Suppose the vehicle is always allowed to turn right, we choose North-South Green, East-West Green, North-South Left Green, East-West Left Green as possible actions. The duration of each phase is 10s.

2.3 Reward

Q-learning will be iteratively trained according to the reward function. The average delay can indicate the congestion of the current intersection. Therefore, the average delay is used as the evaluation measure for the reward.

2.4 ε -greedy

Q-learning will select the action with the largest Q-values as the current action to be performed. To avoid the algorithm falling into a local optimum, it is necessary to choose between exploring new actions and adopting the current optimal action. We implement ε -greedy exploration policy, which selects a random action with a probability ε and selects the action with the highest value with a probability $1-\varepsilon$. The value of ε decreases as training epochs progress according to (2.1).

$$\varepsilon = \max\left(0.01, 1-\frac{n}{N}\right) \tag{2.1}$$

Where n is the current training epoch and N is the total number of training epochs.

2.5 DNN structure

Deep Q-learning uses a deep neural network structure to represent the value of $Q(s,a \mid \theta)$, where θ represents the weights of the neural network. This section will introduce the specific network structure. As mentioned above, the state of the intersection is POMDP. Define the observation state in POMDP as O, then O and real state S are different, that is, $Q(s,a \mid \theta)! = Q(o,a \mid \theta)$.

DRQN has been proven to handle POMDP problems very well [12], so we introduce the DRQN in this work. DRQN is a combination of a Long Short-Term Memory (LSTM) and a Deep Q-Network. LSTM can remember timeline information, by remembering the input states of the previous moments, not just the current moment, the input state of the current intersection can be represented as completely as possible, and the gap between $Q(s,a \mid \theta)$ and $Q(o,a \mid \theta)$ can be narrowed.

Next, we will introduce the network structure of DRQN. The state matrix is fed to a convolutional neural network with two convolutional layers and the activation function uses ReLU, next is the LSTM layer, the activation function also applies ReLU. The number of layers and the number of neurons in the LSTM layer need to be determined by experiment. The final output layer is a fully connected layer, which is used to output the Q-values of all the actions. The network structure is shown in Fig. 3.

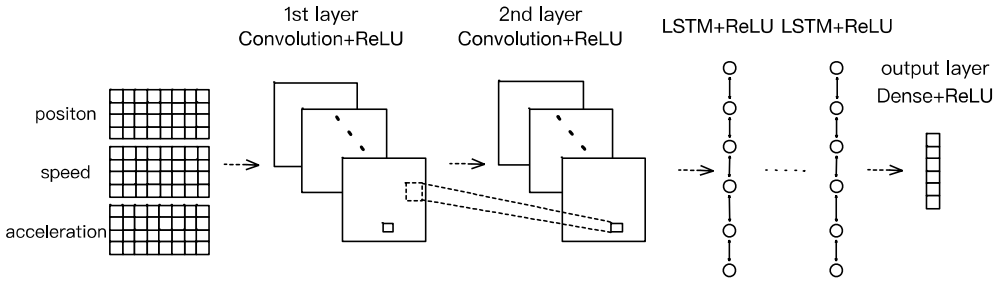

Fig. 3. DRQN network structure

As mentioned above, Q-learning generates many group records (s_t, a_t, r_t, s_{t+1}) in the process of state transition. According to these records, training samples of the deep neural network are constructed. The neural network input is s_t and the output is the target Q value:

$$y_t = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1} \mid \theta) \tag{2.2}$$

Where γ is the discount factor.

The agent learns weights θ by training the DNN network to minimize the following mean squared error (MSE):

$$L(\theta) = E[(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1} \mid \theta) - Q(s_t, a_t \mid \theta))^2] \tag{2.3}$$

We adopt the stochastic gradient descent algorithm $Adam$ to train the network.

To improve algorithm stability, we adopt two methods proposed in [13]: experience replay and the target network.

Experience replay sets a replay memory M to store all records. The replay memory is of finite capacity and when it is full, the oldest data will be discarded. When the agent trains the DNN network, it randomly draws a $batch$ ($batch < M$) size samples from the replay memory M to form input data and target pairs, and then uses these input data and targets to update DNN parameters θ by $Adam$ algorithm.

The structure of the target network is the same as that of the DRQN. It is used to generate the target value of equation (2.2).

Assume that the original network parameter is θ, the target network parameter is θ', θ is updated every step, and θ' is updated to the value of θ after a certain number of steps.

The algorithm is described as follows:

Algorithm 1 Deep reinforcement learning algorithm with experience replay and target network for a single intersection

Initialize the DRQN network structure with weights θ .Initialize target network with weights $\theta^{'} = \theta$
Initialize ε, γ, N
for episode 1 to N **do**
  Initialize intersection state s_0 and action a_0
  for 1 to T **do**
    Select $a_t = \arg\max_a Q(s_t, a \mid \theta)$ with the probability of $1-\varepsilon$,and randomly select an action with the probability of ε
    Perform selected action a_t , get the reward r_t and next state s_{t+1}.
    Store observed experience (s_t, a_t, r_t, s_{t+1}) into replay memory M .
    if the number of samples in the replay memory > *batch* **then**
      Randomly draw a *batch* size samples from memory M ,for each record, $y = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1} \mid \theta^{'})$
      Update θ by applying *Adam* algorithm to training data.
      Update $\theta^{'}$ after reaching a certain number of steps.
    end if
    $s_{t+1} = s_t$
  end for
end for

3 Area Traffic Coordination Control

As discussed above, when dealing with single-intersection problems, we can train the deep Q network so that the agent can choose the best action. When dealing with the area traffic coordination control, which is a multi-agent Q-learning problem, our goal is also to select the joint action that allows multiple intersections to achieve optimal results. A simple idea is that we can choose the optimal actions for each intersection separately, and then combine these actions into a joint action. However, the behavior of the agent i can be regarded as the environment in which the agent j is located. That is to say, the agent i must consider the action of the agent j when selecting the optimal action. Therefore, the optimal action of a multi-agent must be considered together.

Coordination algorithm can be used to obtain multi-agent optimal joint action, commonly used coordination algorithms include variable elimination and Max-plus [14] algorithms. By maximizing over one factor at a time, variable elimination is much faster than optimizing over the entire joint action space. However, it is computationally expensive, as the execution time is exponential as the number of agents increases. In such cases, the Max-plus algorithm is a better choice.

3.1 Max-plus

Suppose that we have a coordination graph $G = (V, E)$, each agent i repeatedly sends a message μ_{ij} to its neighbors j. The message μ_{ij} is defined as:

$$\mu_{ij}(a_j) = \max_{a_i} \left\{ f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \right\} \tag{3.1}$$

where $\Gamma(i) \setminus j$ represents all neighbors of agent i except agent j.

The meaning of this formula is, i send a message to j that consists of a maximization over i and j's factor and the messages i has received from its neighbours that are not j. An agent i only has to sum over the received messages from its neighbours which are defined over individual actions, instead of enumerating over all possible action combinations of its neighbours. Messages are exchanged until they converge to a fixed point, or until some external signal is received. When the iteration of the max-plus algorithm is terminated, each agent i can get the optimal action according to the following equation:

$$a_i^* = \arg\max_{a_i} \sum_{j \in \Gamma(i)} \mu_{ji}(a_i) \tag{3.2}$$

The max-plus algorithm is shown in Algorithm 2.

Algorithm 2 Max-plus algorithm

1: Initialize $\mu_{ij} = \mu_{ji} = 0$ for all vertices (i,j).
2: Initialize fixed point = false, number of iterations $n = 0$, maximum number of iterations are, the threshold Δ
3: **while** fixed point = false and $n < N_{\max}$ **do**
4: fixed point = True, $n = n + 1$.
5: **for** every agent i **do**
6: **for** all neighbors $j = \Gamma(i)$ **do**
7: i send a message $\mu_{ij}(a_j)$ to j according to equation(3.1)
8: **if** $\mu_{ij}(a_j)$ differs from the previous message by Δ **then**
9: fixed point = False
10: **end if**
11: **end for**
12: **end for**
13: **end while**
14: Get the best action a_i^* according to equation (3.2).

3.2 Transfer learning

When dealing with area coordination control problems, it will take a long time if Q-learning is performed for every intersection. According to the literature [15], intersections of same traffic flow are divided into one area, so we can think that the Q functions of all intersections are relatively close. Therefore, we can use transfer learning [16]. Although the approximation of the Q-values is less accurate, the advantage of using transfer learning instead of multi-agent reinforcement learning is that less time has to be spent on training.

3.3 Area coordination control

In this section, we will combine the max-plus algorithm and transfer planning and apply it to area coordination control. In area coordination control, we define $f_{ij}(a_i, a_j)$ as:

$$f_{ij}(a_i, a_j) = Q_i(s_i, a_i \mid \theta_i) + Q_j(s_j, a_j \mid \theta_j) \tag{3.3}$$

Where $Q_i(s_i, a_i \mid \theta_i)$ and $Q_j(s_j, a_j \mid \theta_j)$ are the Q functions for i and j, respectively.

For the pseudocode of the entire algorithm, see Algorithm 3.

Algorithm 3 Area Coordination Control

1: Initialize $Q(s,a \mid \theta)$ for a single intersection using Algorithm 1.
2: Initialize $s = s_0$
3: **for** time step t **do**
4: Get the best action a^* from s according to Algorithm 2.
5: **for** every agent i **do**
6: take action a_i^*
7: **end for**
8: Transfer to a new state s' , $s = s'$
9: **end for**

4 Simulation Results and Analysis

4.1 Simulation settings

In order to validate the performance of DRQN in this paper, microscopic traffic simulation software (USTCMTS2.1) is used, which is developed by the laboratory of the data fusion & intelligent traffic system, University of Science and Technology of China. The DNN network shown in Fig. 3 is implemented using Keras Python libraries.USTCMTS2.1 uses C# as a programming language, C# and Python communicate using the IronPython library. Detailed simulation settings are as follows.

Parameter settings: The agent is trained for $N = 200$ episodes. Each episode lasts $T = 0.1$ hours. Set discount factor $\gamma = 0.95$, the capacity of replay memory $M = 2000$ and $batch = 32$ .Set $N_{\max} = 30$, threshold $\Delta=10^{-9}$.

DNN Structure: The first convolution layer has 32 convolution kernels and the convolution kernel is 3*3. The second convolution layer has 64 convolution kernels and the convolution kernel is also 3*3. Learning rate of DNN is 0.001.

Intersection: As shown in Fig. 4, a typical area consisting of the four intersections is selected. Set road segment L to be 224 meters, cell length C to be 7 meters.

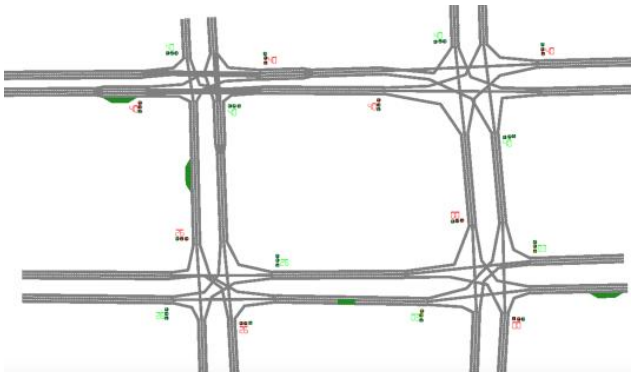

Fig. 4. An area of four intersections

Intersection traffic flow settings: Select three typical traffic flows, low saturation, near saturation, and supersaturation, each intersection has the same flow, flow values are shown in Table 2.

Table 2. Traffic flow

	Road a	Road b	Road c	Road d
Low saturation (veh/h)	96	228	80	236
Near saturation (veh/h)	592	672	352	736
Supersaturation (veh/h)	664	1600	696	1632

4.2 Simulation results

First, we determine the number of LSTM layers and the number of neurons for the above three typical flows, respectively. According to the experience of parameter setting in deep learning, we choose the number of layers of the network from 1 to 3, and the number of neurons is {64, 128, 256, 512}. Using grid search, the average delays of the three types of traffic under different network parameters are shown in Table 3:

Table 3. Average delay under different network parameters (low saturation/near saturation/supersaturation)

	64	128	256	512
1	45.16/66.3/93.12	47.33/59.87/96.1	47.02/61.23/95.33	49.91/65.29/97.72
2	48.67/75.54/89.57	**43.39**/65.38/99.49	50.41/57.81/103.5	53.4/63.74/101.69
3	54.87/62.47/94.78	44.57/64.93/**87.94**	46.14/**56.12**/91.38	45.66/65.18/106.96

As can be seen from the table, the optimal network structure for the three saturations are 2/3/3 LSTM layer with 128/256/128 neurons per layer, respectively.

Then, we analyze the performance of the DRQN proposed in this paper. The network adopts the optimal network structure shown in Table 3. We compare the DRQN algorithm with traditional Q learning and fixed time control. During the running of these algorithms, we record the delay every 180 simulation seconds. Based on these data we can draw the curves of the average delay over time. Since the curves under the three saturation flows are similar, the near saturation flow is taken as an example. Comparison of the average delay over time of three algorithms is shown in Fig. 5.

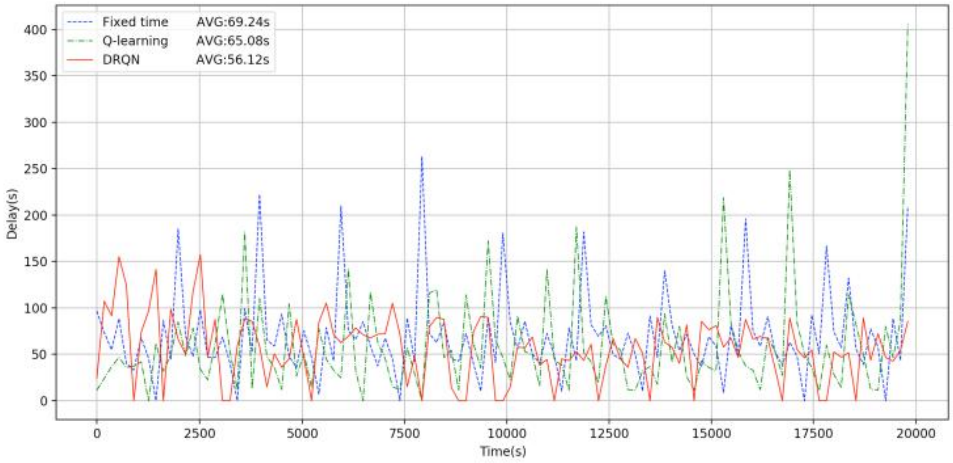

Fig. 5. Average delays over time of the three algorithms

Fig. 5 shows the comparison among DRQN, traditional Q-learning and fixed time control. The average delays for these three algorithms are 56.12s, 65.08s and 69.24s. Compared with Q-learning and fixed time

control, DRQN reduces the average delay by 13.8% and 18.9%, respectively. According to the figure, we can also see that at the beginning of the DRQN process, the average delay is relatively large. This is because the network has not yet converged. As time progresses, the average delay gradually stabilizes. At the same time, the vibration of fixed time control and Q learning is obvious, while the DRQN is relatively stable.

We also run the original DQN algorithm. The average delay for the original DQN is 60.22s. The average delay of DRQN is found to be 6.8% lower than the original DQN's. The result shows that the historical information recorded by the recurrent neural network does make the observed state more in accord with the real intersection state so that the algorithm makes a better choice of action at each time point.

The average delays of the other two saturation flows are shown in Table 4:

Table 4. Low saturation and supersaturation average delay comparison results

	Fixed time(s)	Q-learning(s)	DQN(s)	DRQN(s)
Low saturation	49.7	46.67	44.92	43.39
Supersaturation	109.59	102.25	95.98	87.94

It is found that when two another saturations are taken, the results are similar to that of near saturation. The average delay of the original DQN is smaller than the average delay of fixed time control and Q-learning. The average delay of the DRQN algorithm is also smaller than the original DQN. When the traffic flow is at low saturation, the DRQN algorithm's optimization effect is not so good, because when the traffic flow is small, the vehicle is not so congested during driving, therefore the effect of the algorithm is not obvious. However, when the traffic flow is near saturation and oversaturation, the DRQN algorithm can observe the state of the intersection at each moment to make the best timing choice, thereby improving the traffic efficiency.

5 Conclusion

In this paper, we use DRQN network to solve the intersection signal control problem. At the same time, multi-agent coordination control is implemented using transfer learning and max-plus. Experiments show that the DRQN algorithm improves the performance of DQN algorithm at different saturation flow conditions, and is also superior to fixed time control and Q-learning algorithms. Although we have adopted transfer learning to reduce the training time for Q-learning, it is expected that there will be more methods to be proposed to speed up the algorithm while ensuring the accuracy in the future.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Zhang W, Yu L. Regional intersection coordination control based on BP neural network [J]. Technology & Economy in Areas of Communications. 2016;18(1):38-41.

[2] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning [C]. Proceedings of Workshops at the 26th Neural Information Processing Systems 2013. Lake Tahoe, USA. 2013;201-220.

[3] Ritcher S. Traffic light scheduling using policy-gradient reinforcement learning[C]. The International Conference on Automated Planning and Scheduling. ICAPS; 2007.

[4] Genders W, Razavi S. Using a deep reinforcement learning agent for traffic signal control [J]. arXiv preprint arXiv:1611.01142; 2016.

[5] Gao J, Shen Y, Liu J, et al. Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network [J]. arXiv preprint arXiv:1705.02755; 2017.

[6] Van der Pol E, Oliehoek F A. Coordinated deep reinforcement learners for traffic light control[C]. In Proceedings of NIPS. 2016;16.

[7] Mousavi SS, Schukat M, Howley E. Traffic light control using deep policy-gradient and value-function-based reinforcement learning [J]. Iet Intelligent Transport Systems. 2017;11(7):417-423.

[8] El-Tantawy S, Abdulhai B. Towards multi-agent reinforcement learning for integrated network of optimal traffic controllers (MARLIN-OTC) [J]. Transportation Letters. 2010;2(2):89-110.

[9] El-Tantawy S, Abdulhai B, Abdelgawad H. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on Downtown Toronto [J]. IEEE Transactions on Intelligent Transportation Systems. 2013;14(3):1140-1150.

[10] Zhu F, Aziz HMA, Qian X, et al. A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multi-agent framework [J]. Transportation Research Part C. 2015; 58:487-501.

[11] Abdoos M, Mozayani N, Bazzan AL. Hierarchical control of traffic signals unisg Q-learning with tile coding [J]. Applied Intelligence. 2014;40(2):201-213.

[12] Hausknecht M, Stone P. Deep recurrent Q-learning for partially observable MDPs [C]. Proceedings of the AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents. Arlington: AAAI; 2015.

[13] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature. 2015;518(7540):529-533.

[14] Kuyer L, Whiteson S, Bakker B, et al. Multiagent reinforcement learning for urban traffic control using coordination graphs [C]. European Conference on Machine Learning and Knowledge Discovery in Databases. Springer-Verlag. 2008;656-671.

[15] Zhu L, Chen F. Area delay modeling based on traffic assignment [C]. International Conference on Electrical, Mechanical and Industrial Engineering; 2016.

[16] Van der Pol E. Deep reinforcement learning for coordination in traffic light control [D]. Master's thesis, University of Amsterdam; 2016.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://www.sciencedomain.org/review-history/24643