Received 30 September 2022, accepted 4 November 2022, date of publication 28 November 2022, date of current version 14 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3225431

RESEARCH ARTICLE

Traffic Signal Control System Using Deep Reinforcement Learning With Emphasis on Reinforcing Successful Experiences

NAOKI KODAMA[1], TAKU HARADA[2], AND KAZUTERU MIYAZAKI[3], (Member, IEEE)
[1]Department of Computer Science, School of Science and Technology, Meiji University, Tama-ku, Kawasaki, Kanagawa 214-8571, Japan
[2]Department of Industrial Administration, Faculty of Science and Technology, Tokyo University of Science, Noda-shi, Chiba 278-8510, Japan
[3]National Institution for Academic Degrees and Quality Enhancement of Higher Education, Kodaira-shi, Tokyo 187-8587, Japan

Corresponding author: Naoki Kodama (kodama@meiji.ac.jp)

This work was supported by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research under Grant JP21K12024.

ABSTRACT In recent years, several studies have been conducted on the dynamic control of traffic signal durations using deep reinforcement learning with the aim of reducing traffic congestion. The unique advantages of independent control of traffic signals include reduction in the cost of information transmission and stable control without being affected by the failure of other traffic signals. However, conventional deep reinforcement learning methods such as Deep Q-Network may degrade the learning performance in a multi-agent environment where there are multiple traffic signals in the environment. Therefore, we propose a traffic light control system based on the dual targeting algorithm, which incorporates reinforcement of successful experiences in multi-agent environments, with the aim of realizing a better traffic light control system. The experimental results in a multi-agent environment using a traffic flow simulator based on simulation of urban mobility (SUMO) show that the proposed traffic light control system reduces the waiting time at traffic lights by 33% compared to a conventional traffic light control system using deep reinforcement learning. In the future works, we aim to apply this research to traffic light control systems in real environments.

INDEX TERMS Deep reinforcement learning, deep Q-network, Q-learning, multi-agent, traffic signal control.

I. INTRODUCTION

In recent years, there has been a significant rise in the overall economic losses owing to increasing traffic congestion. For instance, the total time lost owing to traffic congestion in Japan in 2012 was approximately 5 billion hours per year [1]. The economic loss in terms of total lost time is estimated to be approximately 10 trillion yen per year, and it is calculated by converting cash wages in 2012 to time [2]. Therefore, because the economic benefits of reducing traffic congestion are significant, it is necessary to address this issue.

We aim to respond to changes in the environment dynamically using reinforcement learning. However, because traffic congestion is caused by a variety of factors such as events, weather, and time, and uncertain factors such as sudden congestion, designing a control law that considers all of these factors is challenging. Therefore, there is a need to realize a traffic signal control system (TSCS) that automatically obtains a better control law considering multiple factors using deep reinforcement learning.

In recent years, considerable research has been conducted on the application of deep reinforcement learning to TSCSs. For instance, in an environment with a single intersection, a TSCS using a deep Q-network (DQN) [3] has been used to reduce traffic congestion [4], [5], [6]. However, in real environments where there are multiple intersections, TSCSs are expected to interact with each other. Therefore, it is necessary to consider a multi-agent environment where multiple

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

intersections and TSCSs exist. TSCSs with reinforcement learning in a multi-agent environment can be classified into three types [7].

The first is centralized control implementation [8]. In this method, a single control system operates multiple traffic signals. A single control system makes it possible to formalize the problem as a single-agent environment, and the advantage of this system is that the reinforcement learning system for a single agent can be used effectively. However, the transfer cost of collecting information on various traffic signals is high, and the time required for learning depends on the number of actions that can be selected. Additionally, no traffic signals can be controlled if the single control system fails, and the monotonous behavior of the system, such as control by fixed time cycles, can be expected.

The other two are implemented using a distributed control scheme and can be defined according to their implementation method.

In the first method, each intersection exchanges information with the surrounding intersections and then controls the traffic signals placed at each intersection [9], [10]. This method effectively uses deep reinforcement learning to obtain a better control law for the entire system. However, there are some problems in implementing this method in a real environment. It is assumed that the information exchanged with each intersection varies depending on the environment, and the transfer cost of exchanging information with each intersection is high. Additionally, for centralized control, problems such as the impact of failures of surrounding traffic signals can also be anticipated.

In contrast, in the second method, each intersection controls its traffic signals completely independently, as is done previously. In this method, the TSCS is trained using deep reinforcement learning based on the information within each intersection. However, it is difficult to obtain a better control law for the entire system because there is no information transfer between intersections. However, there is no transfer cost, making this the easiest method to implement in a real environment.

In this study, we propose a TSCS that can function without direct communication or transfer cost, and we aim to improve the learning performance of our TSCS. First, the signal control system in this study perceives the change in congestion on a road within the intersection every hour. This observation helps to predict the behavior of traffic signals in the vicinity without the need for information transfer between tne intersections. However, this problem becomes a multi-agent problem, and when dealing with reinforcement learning, the learning performance may be adversely affected by the simultaneous learning problem [11]. Therefore, we use the dual targeting algorithm (DTA) [12] to reduce the impact of the simultaneous learning problem. Compared to DQN, the DTA is a deep reinforcement learning method that places a higher emphasis on the success experience, and it has shown to improve learning performance when implemented in a home energy management system [13]. It has been experimentally confirmed that a method that strongly reinforces the success experience, such as Profit Sharing, can be effectively used to address the simultaneous learning problem [14], and this study aims to achieve this effect through DTA. The TSCS was tested using simulation of urban mobility (SUMO) [15], an open source traffic flow simulator developed by the German Aerospace Center, and we compared this model with TSCSs that use the existing deep reinforcement learning methods.

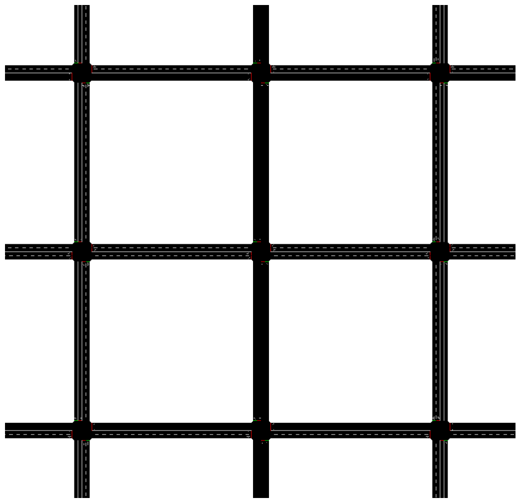

FIGURE 1. **Road network environment. There are nine intersections in the environment, all of which contain traffic light agents.**

The TSCS proposed in this study has the following characteristics:

- No information is transmitted between traffic signals.
- Predicts information on adjacent signals by using changes in congestion within an intersection

Under the TSCS, the following contributions can be obtained by using DTA.

- The number of trials and errors required for learning convergence does not increase.
- Reduced latency by 33% in the learned model.
- The least variation in learning results per experiment.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this study, we assume a road network environment that consists of three one-lane dual carriageways connecting the east-west and north-south directions, as shown in Fig. 1. As shown in Fig. 2, each intersection is equipped with traffic signals, wherein the inner lane is dedicated to left turns and the outer lane is dedicated to straight or right turns. All vehicles appear with a certain probability on each road leading from outside the road network and take the shortest path to their destination. Vehicle destinations are randomly set from the roads exiting the network in the direction opposite to where they appear. For example, a vehicle that appears at the

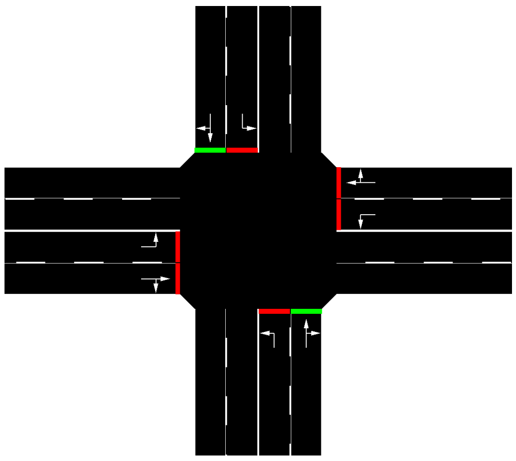

FIGURE 2. Enlarged view of the intersection. The road consists of two lanes on each side.

TABLE 1. Signal phases handled in this environment.

Phase number	E-W OUT	E-W IN	N-S OUT	N-S IN
1	G	R	R	R
2	R	G	R	R
3	R	R	G	R
4	R	R	R	G

west end of the network sets one of the roads at the east end as its destination. Therefore, all vehicles will pass through at least three intersections. When a vehicle passes through an intersection, it follows four signal phases, as listed in Table 1. Here, E-W and N-S denote east-west and north-south roads, respectively. If the word OUT appears after E-W or N-S, it means the outer lane of E-W or N-S. If the word IN appears after E-W or N-S, it means the inner lane of E-W or N-S. The green and red signals for is lane are denoted by G and R, respectively When the traffic signal phase switches, the yellow light is turned on for a certain period between the green and red signal phases for safety purposes.

In our TSCS, deep reinforcement learning was used under discretized time steps $t = 1, 2, \cdots$. At each step, some traffic information observed from the environment is obtained as a state, and each signal switches between signal phases based on the state. Our TSCS aims to reduce traffic congestion across the entire road network and improve its strategies based on the rewards received from the environment. In this section, we describe a deep reinforcement-learning model for our TSCS.

A. STATE DEFINITION

At each intersection, the state is defined at each step t using the phase P_t of the traffic signal, the congestion level D_t^l of lane l leading to the intersection, the change in congestion level ΔD_t^l in one step, and the average speed V_t^l of the vehicles on lane l leading to the intersection.

The phases of a traffic signal are represented using a one-hot representation.

D_t^l is in the range [0, 1]; it is 1 when a vehicle, up to the maximum occupancy of the road, is in the lane and 0 when no vehicle is in the lane. Therefore, D_t^l is calculated as follows.

$$D_t^l = \frac{N_{\mathrm{veh}}^l (S_{\mathrm{veh}} + g_{\mathrm{min}})}{L^l} \tag{1}$$

where N_{veh}^l is the number of vehicles in lane l, L^l is the length of lane l, S_{veh} is the size of the vehicle and g is the minimum distance between vehicles. The congestion level is used to determine whether each lane is congested. For example, a lane with a high congestion level is congested. Therefore, in such a case, it is necessary to switch to a traffic signal phase that allows traffic to pass to alleviate congestion.

Because ΔD_t^l is the change in D_t^l during one step, it can be calculated by taking the difference between D_t^l and D_{t-1}^l. In this case, ΔD_t^l can take the range $[-1, 1]$; however, in this study, ΔD_t^l is used after normalization. Therefore, ΔD_t^l is calculated as follows.

$$\Delta D_t^l = \frac{1 + D_t^l - D_{t-1}^l}{2} \tag{2}$$

The amount of change in congestion was used to determine the status of adjacent traffic signals. For example, if congestion increases, we know that an intersection other than our own, to which the affected lane is connected, has adopted a signal phase that allows traffic to enter our intersection. In such a case, the traffic signal must also allow traffic in the lane with increasing congestion to ease the congestion. In other words, by simply using the change in congestion level as a state, each traffic signal can generate the possibility of obtaining cooperative control with neighboring traffic signals.

The V_t^l takes the range $[0, V_{\max}]$. Here, $V_{\max}$ represents the maximum speed of the road. In this study, V_t^l was normalized and used; thus, it was calculated as follows:

$$V_t^l = \frac{V_t^l}{V_{\max}} \tag{3}$$

The average speed of the cars in lane l that are heading to an intersection was used to determine the flow of cars at the intersection. For example, even if the congestion level is high, it is not a problem if the average speed is high because cars are passing normally in that lane. This information, which cannot be determined solely by the congestion level, can be obtained using the average speed.

In summary, the traffic signal agent in this environment defines the state at a discrete time step t as $s_t = (P_t, D_t^l, \Delta D_t^l, V_t^l)$.

B. ACTION DEFINITION

Each traffic-light agent selects an action at each step t and observes the state s_{t+1} of the transition destination. In this environment, each agent defines the switching of its signal phase as an action. Specifically, for state s_t, each agent selects one of the four signal phases listed in Table 1 and outputs it as an action. If the selected signal phase is the same as the current signal phase, the current signal phase will be retained. If a signal phase that is different from the current phase is selected, the current signal phase will switch to the chosen signal phase after the yellow signal phase of a certain duration. However, if the same phase continues for the $T_{\max}$ steps, the phase is forcibly switched, ignoring the agent's actions. The phase to switch to is determined by the order of the phase numbers shown in Table 1. For example, if the current phase is phase 3, the phase is switched to phase 4 when the forced phase switch is performed. If the current phase is 4, it switches to phase 1.

C. REWARD DESIGN

The objective of the TSCS in this study is to alleviate traffic congestion in the entire road network. However, because each traffic signal agent should obtain traffic information for the entire road network, which leads to an increase in the transfer cost, the objective of each agent in this study is to minimize traffic congestion at its own intersection. Although achieving the goal of each agent does not necessarily help achieve the goal of the system as a whole, we anticipate achieving the goal of the system as a whole through cooperative control. Cooperative control can be obtained by sharing information with neighboring traffic signals through state perception.

In this study, we used the accumulated waiting time of vehicles as an indicator of traffic congestion at an intersection [16]. First, we define $w_{i_t,t}$, $(1 \leq i_t \leq N_{\text{veh}})$ as the time (in seconds) until step t of the vehicle, that is, the i_tth vehicle in the lane heading to its own intersection at step t. Here, N_{veh} is the number of vehicles in all lanes heading to their own intersection. The cumulative waiting time W_t at step t is calculated as follows.

$$W_t = \sum_{i_t=1}^{N_{\text{veh}}} w_{i_t,t} \tag{4}$$

W_t does not decrease with the step, and the larger the increase with respect to the step, the larger the number of vehicles waiting at the intersection. In other words, to reduce traffic congestion at an intersection, the smaller the increase or decrease in W_t, the larger the reward. Therefore, the reward at step t can be designed as follows

$$r_t = W_{t-1} - W_t \tag{5}$$

The reward always considers a negative value, and the maximum value is zero when no vehicle is waiting at the intersection. Therefore, by using deep reinforcement learning to learn a strategy that maximizes discounted revenue, congestion relief at the intersection of each agent can be achieved.

III. SIGNAL CONTROL SYSTEM USING DUAL TARGETING ALGORITHM

A. APPROACH OF THIS STUDY

This study assumes an environment that can be realized with a lower transfer cost than that of a conventional traffic signal system by eliminating the communication between traffic signals. However, because multiple traffic signals in the environment mutually affect each other, unstable learning due to simultaneous learning problems becomes an issue when using conventional deep reinforcement learning, such as DQN, to acquire control measurements. Simultaneous learning can lead to a situation where a strategy change by one mutually influencing agent affects the environment of other agents, making learning difficult. It has been experimentally that each agent should converge its strategy quickly in order to address the aforementioned problem, for instance, by using exploitation-oriented learning [14].

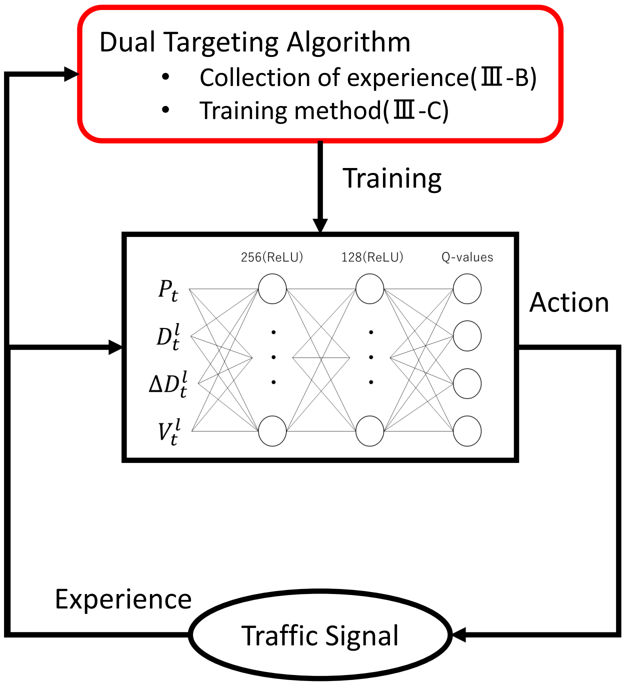

FIGURE 3. Conceptual diagram of proposed traffic light control system.

In this study, we aim to reduce the impact of this simultaneous learning problem by using DTA, which strongly reinforces the success experience for learning a TSCS. Conceptual diagram of traffic light control system is shown in Fig. 3. Furthermore, the neural network architecture handled in this study is shown in fig. 4. To verify the feasibility of DTA, this study also deals with a signal control system wherein the deep reinforcement learning used is replaced by DQN and multistep DQN, and this is further used as a comparison target in the literature [12].

B. COLLECTION OF EXPERIENCE

In the traffic signal system proposed in this research, the experience collection and training are performed according to DTA.

The experience collection is performed according to the following steps. First, in each step, the state s_t is perceived by the TSCS, the action a_t is performed, the reward r_{t+1} is obtained from the environment, and the transition destination state s_{t+1} is stored in the replay buffer. Next, the time when the signal is switched is defined as the end of the episode. At the end of the episode, the multistep returns $R_i^{\tau_i-t+1}$ from steps i to τ_i, the end-of-episode step, and the

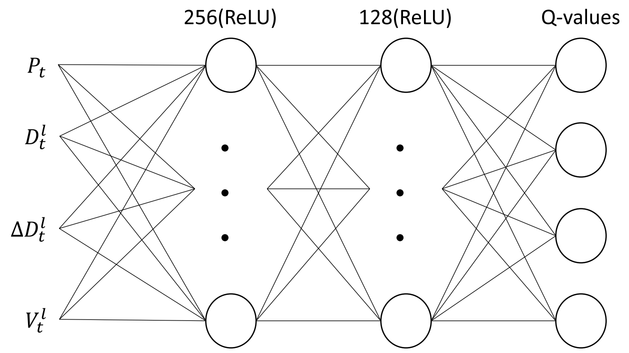

FIGURE 4. A neural network architecture handled by the proposed TSCS. The input is state $s_t = (P_t, D_t^l, \Delta D_t^l, V_t^l)$, the first hidden layer has 256 ReLU units, the second has 128 ReLU units, and the output is Q-values used for action selection.

state s_{t+1} of the next step of the episode are stored in the replay buffer. The state s_{τ_i+1} of the next step of the termination is added to the experience of the corresponding step in the replay buffer. In other words, the experiences in the episode that have not reached the end of the replay buffer keep $(s_t, a_t, r_{t+1}, s_{t+1})$, and the other experiences keep $(s_t, a_t, r_{t+1}, s_{t+1}, R_i^{\tau_i-t+1}, s_{\tau_i+1})$. Here, $R_i^{\tau_i-t+1}$ is calculated as follows.

$$R_i^{(\tau_i-t+1)} = \sum_{k=0}^{\tau_i-i+1} \gamma^k r_{i+k+1} \tag{6}$$

A certain number of experiences were randomly sampled from the replay memory during training, and the neural network was trained based on these experiences.

First, we calculate the 1-step targets y_j and multistep targets f_j from the sampled step experiences, as follows:

$$y_j = r_{j+1} + \gamma \max_{a'} Q(s_{j+1}, a'|\bar{\theta}), \tag{7}$$

$$f_j = R_j^{(\tau_j-j+1)} + \gamma^{(\tau_j-j+1)} \max_{a'} Q(s_{\tau_j+1}, a'|\bar{\theta}). \tag{8}$$

where j is a randomly sampled experience step, $Q(s, a|\theta)$ is the action value function (Q-value) for action a when state s is input to a neural network with learning weight θ, and $\bar{\theta}$ is the target. The learning weights of the target network [1] are shown. The training weights of the target network were updated by copying from θ at every step.

In DTA, if f_j is a large return, which is successful experience, then the Q value is updated by the return. Therefore, the error is calculated using the maximum of y_j and f_j as the target value as follows.

$$\delta_j(\theta^Q) = \max(y_j, f_j) - Q(s_j, a_j|\theta^Q), \tag{9}$$

The training of the neural network is updated by the gradient descent method using error $L(\theta)$, which is calculated as follows:

$$L(\theta^Q) = \begin{cases} \frac{1}{M}\sum_j \frac{1}{2}\delta_j(\theta^Q)^2 & \text{if } |\delta_j(\theta^Q)| \geq 1.0 \\ \frac{1}{M}\sum_j |\delta_j(\theta^Q)| - \frac{1}{2} & \text{otherwise,} \end{cases} \tag{10}$$

where M denotes the mini-batch size.

TABLE 2. Parameter settings.

Parameter	Value
Batch size M	32
Replay buffer size	50,000
Training start step	1,000
Target network update frequency	1,000
discount factor γ	0.8
Optimizer	Adam [17]
Learning rate for adam	0.000625
Strategy method when training	ϵ-greedy($\epsilon = 0.05$)
Strategy method when evaluation	greedy

This algorithm is summarized in Algorithm 1.

IV. EVALUATION

A. SIMULATION SETTING

In this experiment, we verified the learning performance of the proposed method on a TSCS without information transfer between intersections. For the proposed TSCS, we compared three types of systems: random behavior, a similar TSCS learned using DQN, and a similar traffic signal system learned using multistep DQN, which was selected as a comparison target in the literature [1].

The hyperparameters for deep reinforcement learning used in this experiment are listed in Table 2. All parameters not shown here follow the literature [3]. In addition, the number of steps in the multistep DQN was set to three based on preliminary experiments. In this experiment, we used the same neural network architecture for all methods to compare the learning performance of the different learning algorithms. The input layer, wherein the state is used as the input, consists of 28 units. The information in the input layer is weighted by all coupling layers and fed into the 256 ReLUs of the first hidden layer. Then, the output of the first hidden layer was weighted by all the coupling layers and input to the 128 ReLUs of the second hidden layer. Finally, the output of the second hidden layer is the Q-value of the action that selects each phase number.

In this study, Rllib [18] was used to develop the algorithm. Additionally, the simulation of urban mobility (SUMO) [15] was used to construct the simulation environment. As settings for roads, the maximum speed of each road was set to 50[km/h], the vehicle occurrence probability of each road was set to 0.015 per step, and the distance between intersections was set to 100[m]. As a setting for traffic signals, the duration of the same signal phase is set to 50[s], and if it is exceeded, the current phase number is shifted to the following phase number, ignoring the agent's action selection.

In this experiment, one step was defined as 5 s in the simulation, and 360 steps (30 min) were considered one trial. The duration of the yellow signal is set as 2 s. When switching signals, the action for this step is performed as a yellow signal (2 s) + new signal phase (3 s).

B. EVALUATION METHOD

At the end of the trial, the system enters the termination state, and the state is re-initialized. After every 20 training trials, the

Algorithm 1 DTA-Based TSCS

```
Input: The phase $P_t$, the congestion level $D_t^l$, the change in congestion level $\Delta D_t^l$, the average speed $V_t^l$, the step $t$
Output: The weights of a neural network $\theta$
 1: Initialize replay memory $D$ of size $N$, budget $H$, and minibatch $M$
 2: Randomly initialize the neural network $Q(s, a|\theta)$ with weights $\theta$
 3: Initialize the weights of a target network $\theta$ by copying $\theta$
 4: Initialize the time step at the start of the episode $t_{\text{start}} = 0$
 5: Initialize List Rewards for temporarily storing rewards.
 6: Observe the initial state $s_0$
 7: for $t = 0, 1, \cdots, H$ do
 8:     Choose an action $a_t$ and observe a reward $r_{t+1}$ and the next state $s_{t+1}$
 9:     Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $D$
10:     Append the $r_{t+1}$ to Rewards
11:     if $t$ is episode terminal then
12:         for $j = t_{\text{start}}$ to $t$ do
13:             Set $\tau_j = t$
14:             Compute $R_j^{(\tau_j - j+1)} = \sum_{k=0}^{\tau_j - j} \gamma^k r_{j+k+1}$ from Rewards
15:             Add $\tau_j, R_j^{(\tau_j - j+1)}, s_{\tau_j+1})$ to $(s_j, a_j, r_{j+1}, s_{j+1})$ in $D$
16:         end for
17:         Set $t_{\text{start}} = t + 1$
18:     end if
19:     Sample a random minibatch of $M$ transitions $(s_i, a_i, r_{i+1}, s_{i+1}, \tau_i, R_i^{(\tau_i - i+1)}, s_{\tau_i+1})$ from $D$
20:     Set $y_j = r_{j+1} + \gamma \max_{a'} Q(s_{j+1}, a'|\theta)$
21:     Set $f_j = R_j^{(\tau_j - j+1)} + \gamma^{(\tau_j - j+1)} \max_{a'} Q(s_{\tau_j+1}, a'|\theta)$
22:     Update $\theta$ by minimizing the huber loss $L(\theta)$
23:     Update target networks
24: end for
```

system enters the evaluation mode and executes five trials. One experiment consisted of 1,000 trials, and the learning curve was formed using the average of the results in the evaluation mode for one experiment. In addition, the same experiment was performed ten times with different random numbers, and the average of the ten obtained learning curves was used for the evaluation of this experiment. The average waiting time per vehicle at a single intersection is used as the evaluation index.

C. SIMULATION RESULTS

A comparison of the learning curves is shown in Fig. 3. Fig. 4 shows a magnified view after 500 trials. Furthermore, in Fig. 7, Fig. 8, and Fig. 9, the variation of the DQN after 500 trials, the Multistep DQN, and the 10 experiments of the proposed method are shown as error bars. The unbiased sample variance at the 1,000th trial was 16.5 for the DQN, 0.06 for the multistep DQN, and 0.02 for the proposed method, respectively.

To heck whether the difference between the mean values of the waiting times of the Multistep DQN and the proposed method at the 1,000th trial is significant, the null hypothesis H_0 is "the mean value of the waiting time of the proposed method is equal to the mean value of the Multistep DQN" and the alternative hypothesis H_1 is "the mean of the waiting time of the proposed method is less than the mean of the Multistep DQN." A Welch's t-test was conducted with the significance level set at 0.01. The results showed $P = 0.0003 < 0.01$, and the null hypothesis was rejected.

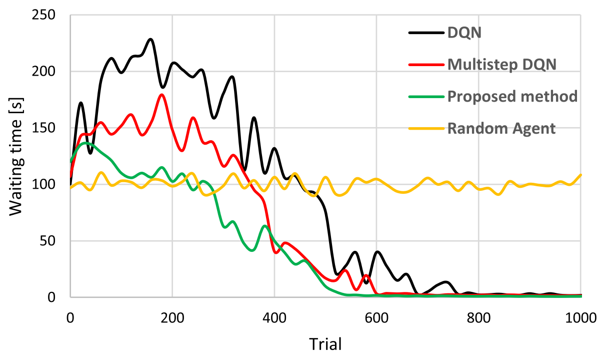

FIGURE 5. Waiting time comparison.

D. CONSIDERATION

From Fig. 3, it is evident that when deep reinforcement learning was introduced, the performance of the proposed method was initially poor, compared to that of the random agent. However, the latency was significantly reduced. We can also confirm that the proposed method using DTA improves the learning speed and converges faster than methods that use

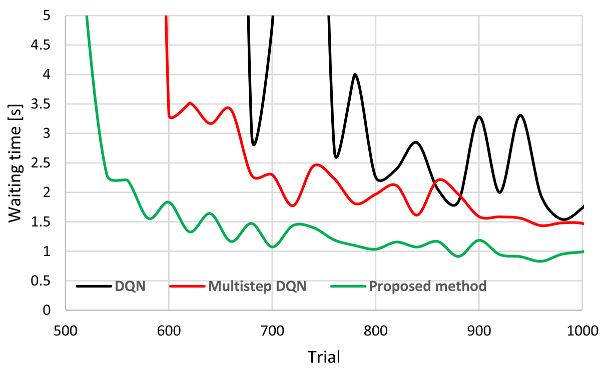

FIGURE 6. Waiting time comparison (enlarged figure after 500 trials).

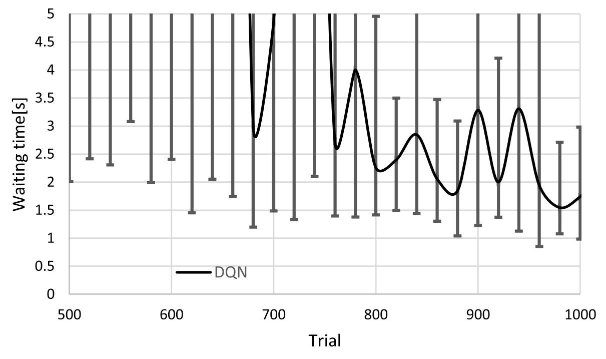

FIGURE 7. Error bars of DQN in 10 experiments.

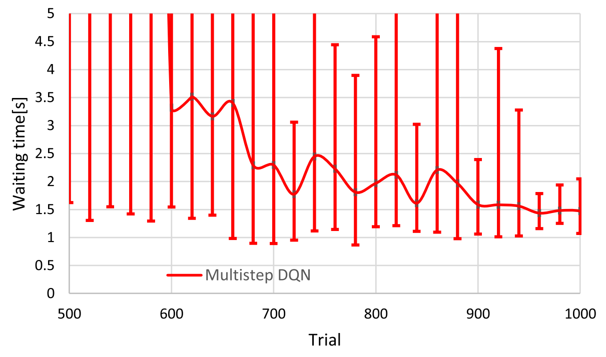

FIGURE 8. Error bars of multistep DQN in 10 experiments.

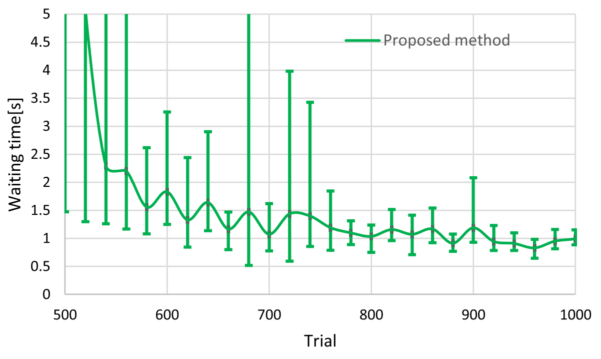

FIGURE 9. Error bars of proposed method in 10 experiments.

DQN and multistep DQN. In addition, because multistep DQN converges more quickly than DQN, these improvements in learning speed can be attributed to the use of multistep returns. In DTA, the learning speed may have increased because DQN takes longer than multistep DQN, depending on the situation.

Fig.4 shows that the proposed method reduces the latency by more than 33% compared with the existing methods. It was also confirmed that the proposed method can stably reduce the waiting time because the unbiased sample variance of the proposed method is the smallest.

Because this is a multi-agent environment, it is expected that the improvement of an agent's strategies may be affected by changes in the strategies of other agents because of the simultaneous learning problem. In other words, because of the impact of the simultaneous learning problem, Q-values may not be estimated correctly. Because the effect of the simultaneous learning problem is reduced when all agents solidify their measures to a certain extent, the multistep DQN and proposed method can immediately reduce its effect due to their high learning speed. In particular, the proposed method strongly reinforces a set of actions when the revenue during one cycle of signals (from one signal switch to the next signal switch) is greater than the current Q-value, enabling immediate learning when the Q-value is incorrectly estimated in the early stages of learning. Therefore, the method is less affected by the simultaneous learning problem than the other methods, leading to stable learning.

These results indicate that the proposed method is the most effective for traffic signal control without information transfer between intersections.

In recent years, there has been significant research on various aspects of the performance of deep reinforcement learning, and combining these research studies could help maximize the performance of the reinforcement learning. The DTA used in the proposed method is modified such that the targets for network update are calculated separately from those of the base deep reinforcement learning method to ensure it can be easily merged with other deep reinforcement learning methods. However, compared to existing methods, the proposed method increases the volume of information stored in the replay buffer and the computation of target values for neural network training. Although this increase can be ignored on a large-scale computer, the increase in computation time and memory shortage are on a small computer that is expected to be installed at each traffic light is highly significant. Therefore, for the use of the proposed method in a real-world environment, the computational cost of possible implementations should also be considered.

V. CONCLUSION

In this study, we propose a TSCS that can learn effectively without information transfer between traffic signals to reduce the transfer cost. Traffic flow simulation results show that the proposed method reduces the waiting time by more than 33% compared with similar TSCSs using DQN and multistep DQN. The learning time to convergence is also shorter than that of the existing methods, indicating that the method is effective even when real-time learning is considered. However, on a small computer such as those expected to be installed at each traffic signal in a real environment, we are concerned about increased computation time and insufficient memory compared to existing methods.

In the future, it is necessary to verify the execution time of the proposed method and the existing method on a small computer. In addition, the traffic flow on a real road network changes depending on the time of the day. In this experiment, the simulation time was 800 s under constant traffic flow; therefore, the learning performance under varying traffic flow has not yet been confirmed. Therefore, it is necessary to verify the proposed system in a more realistic environment in the future.

REFERENCES

[1] *Ministry of Land, Infrastructure, Transport and Tourism, Ministry of Land, Infrastructure, Transport and Tourism Productivity Revolution Project*. [Online]. Available: https://www.mlit.go.jp/common/001132350.pdf

[2] (2012). *Ministry of Health, Labour and Welfare, Basic Survey on Wage Structure*. Accessed: Feb. 24, 2022. [Online]. Available: https://www.mhlw.go.jp/toukei/itiran/roudou/chingin/kouzou/z2012/

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, G. M. Bellemare, A. Graves, M. Riedmiller, K. A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[4] H. Wei, G. Zheng, H. Yao, and Z. L. Intellilight, "A reinforcement learning approach for intelligent traffic signal control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2496–2505.

[5] L. A. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 412–421, Jun. 2011.

[6] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," 2016, *arXiv:1611.01142*.

[7] T. Wu, P. Chou, K. Liu, Y. Yuan, X. Wang, H. Huang, and D. O. Wu, "Multi-agent deep reinforcement learning for urban traffic signal control in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8243–8256, Aug. 2020.

[8] Z. Zhang, J. Yang, and H. Zha, "Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization," 2019, *arXiv:1909.10651*.

[9] M. A. Marco, "Multi-agent reinforcement learning for traffic signal control," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 1151–1158.

[10] K. J. Prabuchandran, H. K. An, and S. Bhatnagar, "Multi-agent reinforcement learning for traffic signal control," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 2529–2534.

[11] S. Sen and M. Sekaran, "Multiagent coordination with learning classifier systems," in *Adaption and Learning in Multi-Agent Systems*. Berlin, Germany: Springer, 1995, pp. 218–233.

[12] N. Kodama, T. Harada, and K. Miyazaki, "Deep reinforcement learning with dual targeting algorithm," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–6.

[13] N. Kodama, T. Harada, and K. Miyazaki, "Home energy management algorithm based on deep reinforcement learning using multistep prediction," *IEEE Access*, vol. 9, pp. 153108–153115, 2021, doi: 10.1109/ACCESS.2021.3126365.

[14] N. Kodama, K. Miyazaki, and H. Kobayashi, "Proposal and evaluation of reward sharing method based on safety level," *SICE J. Control, Meas., Syst. Integr.*, vol. 11, no. 3, pp. 207–213, May 2018, doi: 10.9746/JCMSI.11.207.

[15] P. A. Lopez, E. Wiessner, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, and P. Wagner, "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582.

[16] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," 2017, *arXiv:1705.02755*.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[18] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," 2017, *arXiv:1712.09381*.

NAOKI KODAMA received the B.E. degree in mechanical engineering informatics and the M.E. degree in mechanical engineering from Meiji University, Japan, in 2016 and 2018, respectively, and the Ph.D. degree in industrial administration from the Tokyo University of Science, Japan, in 2021.

He is currently an Assistant Professor at Meiji University. He is involved with research of machine learning, particularly reinforcement learning and deep reinforcement learning.

TAKU HARADA received the Ph.D. degree from the Tokyo University of Science, in 1993. He is currently pursuing the Doctor of Engineering degree.

He was an Assistant Professor with the Tokyo University of Science, an Associate Professor with Mie University, and a Junior Associate Professor with the Tokyo University of Science, where he is currently an Associate Professor. He is involved with research of machine learning and metaheuristic optimization.

KAZUTERU MIYAZAKI (Member, IEEE) received the Graduate degree in precise engineering from the Faculty of Engineering, Meiji University, Japan, in 1981, and the Ph.D. degree from the Department of Computational Intelligence, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, Japan, in 1996.

In 1996, he was worked by the Institute as a Research Associate with the Interdisciplinary Graduate School of Science and Engineering, Japan, and in 1999, he became an Associate Professor at the National Institution for Academic Degrees and Quality Enhancement of Higher Education, Japan, where he is currently a Professor at the Research Department. He is involved with research of machine learning, particularly, reinforcement learning, deep reinforcement learning, and text mining.