2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets

Tomoki Nishi*, Keisuke Otaki, Keiichiro Hayakawa and Takayoshi Yoshimura

Abstract— **Traffic signal control can mitigate traffic congestion and reduce travel time. A model-free reinforcement learning (RL) approach is a powerful framework for learning a responsive traffic control policy for short-term traffic demand changes without prior environmental knowledge. Previous RL approaches could handle high-dimensional feature space using a standard neural network, e.g., a convolutional neural network; however, to control traffic on a road network with multiple intersections, the geometric features between roads had to be created manually. Rather than using manually crafted geometric features, we developed an RL-based traffic signal control method that employs a graph convolutional neural network (GCNN). GCNNs can automatically extract features considering the traffic features between distant roads by stacking multiple neural network layers. We numerically evaluated the proposed method in a six-intersection environment. The results demonstrate that the proposed method can find comparable policies twice as fast as the conventional RL method with a neural network and can adapt to more extensive traffic demand changes.**

I. Introduction

Traffic congestion causes serious problems in urban areas [1], [2], e.g., increased travel time, high fuel consumption, and increase in the number of traffic accidents. Traffic signal control is a practical solution to mitigate congestion without limiting human behavior. Typically, traffic signals are controlled using offline-tuned fixed timing period collected from real-world traffic data. However, such traffic signals cannot efficiently respond to the short-term changes in traffic demands.

Adaptive traffic signal control methods that realize responsive traffic control have been developed using genetic algorithms [3], swarm intelligence [4], neural networks [5], mixed integer linear program formulation [6], and reinforcement learning (RL) [7], [8]. We employ model-free RL to find policies to control traffic signals without prior knowledge about the given environment.

Recent traffic signal control approaches based on RL with a neural network, e.g., a convolutional neural network (CNN) have been developed [9], [10]. To extract abstracted features from data, tensor form input data, e.g. a vector or a matrix, is necessary for standard neural networks. Adjacency relations represented with grid graphs like images are additionally required for CNNs. Such representations, however, are not often suited to treat traffic information in road networks because its information is generally represented by values on directed graphs. We illustrate a snapshot of traffic in a four-way intersection in Figure 1 (a). Figure 1 (b)-(d) show typical feature representations for the traffic. Vectors cannot directly handle geometric features of the road network. We need to manually design the futures using a neural network structure, e.g., links between layers. Matrices are available to utilize those features but spend many wasted elements for non-road areas. Directed graphs are efficient to handle the features but is not available for CNNs.

Graph convolutional neural networks (GCNNs) can handle input (e.g., signals or feature values) given on general graphs [11], [12]. Recently, several computationally efficient GCNNs have been proposed [13], [14]. Schlichtkrull et al. [13] proposed a GCNN that can be applied to relational graphs. Their method can extract features in considering the distant vertices on a graph by stacking additional layers. They demonstrated that their network could extract useful features for several domains, e.g., link prediction in relational graphs.

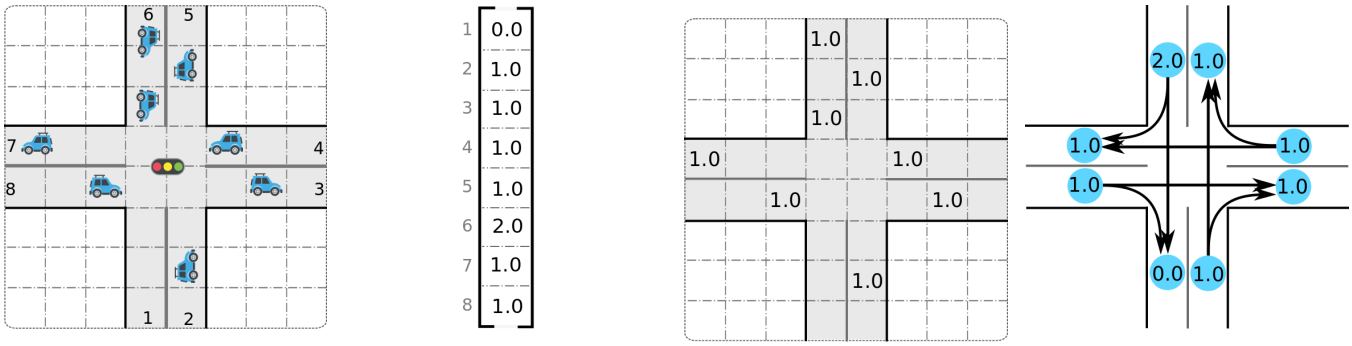
Herein, we developed an RL-based traffic signal control method with a GCNN to handle traffic on a road network with multiple intersections. The proposed approach uses the GCNN to directly extract geometric road network features and adaptively learns a policy using a model-free RL method. We used neural fitted Q-iteration (NFQI), a batch model-free RL method [15], because this method can find a policy from the data collected in advance rather than exploring policies in the real world. NFQI does not require optimistic exploration led to traffic congestion. Each agent learns a policy to control a traffic signal at an intersection in a distributed manner using the traffic information of peripheral roads rather than communicating with other agents. We numerically evaluated the proposed method in a six-intersection environment using the Simulation of Urban Mobility (SUMO) simulation suite [16], a well-known traffic simulator, to validate its efficiency. The simulation results demonstrate that the proposed method can find comparable policies twice as fast as the RL-based method with an fully-connected neural networks (FCNN) and handle broader changes in traffic demand.

The primary contributions of this study are as follows.

- We develop an RL-based traffic signal control method that uses a GCNN to automatically extract the traffic features of peripheral roads in a road network with multiple intersections.
- Agents are implemented in a distributed manner to learn a policy to control a traffic signal at an intersection using the proposed method rather than communicating with other agents directly.
- We demonstrate that the proposed method can find

Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa and Takayoshi Yoshimura are with Toyota Central R&D Labs., Inc.,41-1 Yokomichi, Nagakute, Aichi, Japan
* Correspoinding author, Email: nishi@mosk.tytlabs.co.jp

978-1-7281-0323-5/18/$31.00 ©2018 IEEE



(a) Snapshot of traffic on a four-way intersection

(b) Vector with the number of vehicles per lane

(c) Matrix with vehicle positions

(d) Directed graph with the number of vehicles per lane

Fig. 1: Snapshot of traffic in a four-way intersection (a), and feature representations for the traffic: (b) a vector with the number of vehicles per lane, (c) a matrix with vehicle positions, where blanks are filled with 0.0, and (d) directed graph with the number of vehicles per lane.

comparable policies twice as fast as the conventional RL-based method with a FCNN and can cope with broader changes in traffic demand.

The remainder of this paper is organized as follows. Section II describes the Markov decision process (MDP), NFQI, and GCNN. Section III describes the proposed traffic signal control method, which is based on NFQI with a GCNN. The evaluation results are presented in Section IV, and conclusions and suggestions for future work are given in Section V.

II. Preliminaries

A. MDP

An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a state transition function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a cost function, and γ is a cost discount factor at the next step. A policy $\pi : \mathcal{S} \to \mathcal{A}$ is a function to return an action $a \in \mathcal{A}$ when observing a state $s \in \mathcal{S}$. Here, an optimal policy, $\pi^* : \mathcal{S} \to \mathcal{A}$, minimizes the expected cumulative cost J, which is expressed as follows:

$$J := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right],$$

where $\mathbb{E}_\pi[\cdot]$ is the expected value under policy π and r_t is an instantaneous cost at time step t.

Under a policy π, we define a state value function V^π, i.e., the V-value function, and a state action value function Q^π, i.e., the Q-value function as follows:

$$V^\pi(s) := \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s \right] \quad s \in \mathcal{S},$$

$$Q^\pi(s, a) := \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a \right] \quad s \in \mathcal{S}, a \in \mathcal{A}.$$

The optimal policy π^*, V-value function $V^*(s)$ under π^*, and Q-value function $Q^*(s, a)$ under π^* satisfy the following Bellman equations:

$$Q^*(s, a) = r(s, a) + \mathbb{E}_{s' \sim p(s'|s, \pi^*(s))} \left[\gamma V^*(s') \right],$$
$$V^*(s) = \min_{a \in \mathcal{A}} Q^*(s, a),$$
$$\pi^*(s) = \arg\min_{a \in \mathcal{A}} Q^*(s, a).$$

A conventional RL method, Q-learning with function approximation, learns a policy via estimating the Q-value function by minimizing the following temporal difference (TD) error $\mathcal{E}$ [17]:

$$\mathcal{E}_t = r(s_t, a_t) + \gamma \min_{a' \in \mathcal{A}} \hat{Q}(s_{t+1}, a'; \boldsymbol{\theta}) - \hat{Q}(s_t, a_t; \boldsymbol{\theta}), \quad (1)$$

where $\hat{Q}$ denotes an approximated Q-value function with parameter $\boldsymbol{\theta}$. The TD error is calculated with the sampled state transition under the current learned policy, and the estimated Q-value function is updated toward the TD error descent with the partial derivative of the TD error w.r.t $\boldsymbol{\theta}$.

B. NFQI

Batch RL algorithms are applicable to learn policies from collected data in advance and FQI [18] is a popular batch RL algorithm, which updates an estimated Q-value function $\hat{Q}$ in nearly the same manner as Q-learning; however, FQI uses the data collected according to a predetermined policy rather than a learned policy.

Given a collected dataset $\mathcal{D} = \{\langle s_t, a_t, r_t, s_{t+1} \rangle_{t=1}^{T_i}\}_{i=1}^{D}$, consisting of D length T_i sequence of a state, an action, a cost, and a next state, named *trajectory*, an estimated Q-value function $\hat{Q}$ is updated as follows:

1) Choose the i-th trajectory in $\mathcal{D}$ uniformly at random.
2) Calculate the TD error $\mathcal{E}_t$ using Eq. (1), where t is uniformly drawn at random from a set $\{1, \ldots, T_i\}$.
3) Update the estimated Q-value function $\hat{Q}$ toward the TD error descent with the partial derivative of $\mathcal{E}_t$.

We use NFQI to approximate the Q-value function using a neural network [15]. The neural network weights are updated by propagating the derivative of the TD error in a backward

manner, in step 3 above. We also use a multi-step approach to update weights with the calculated TD errors because previous studies have indicated that such an approach can reduce learning variance [19], [20]. The k-step NFQI is shown in Algorithm 1.

Algorithm 1 k-step Neural Fitted Q-Iteration (NFQI)

Input: An estimated Q-value function $\hat{Q}$ and a set $\mathcal{D}$ of collected trajectories.
Output: The updated Q-value function $\hat{Q}$

1: **procedure**
2: Initialize partial derivative of the weights: $d\boldsymbol{\theta} \leftarrow \mathbf{0}$
3: Choose i-th trajectory D_i uniformly at random from $\mathcal{D}$
4: Draw the final time step t_f uniformly at random from the set $\{k+1, \ldots, T_i\}$
5: $R \leftarrow \min_a \hat{Q}(s_{t_f+1}, a; \boldsymbol{\theta})$
6: **for** step $t = t_f$ to $t_f - k$ **do**
7: $R \leftarrow r_t + \gamma R$
8: Calculate TD error $\mathcal{E}_t$: $\mathcal{E}_t \leftarrow \left(R - \hat{Q}(s_t, a_t; \boldsymbol{\theta})\right)$
9: Calculate partial derivative of $\mathcal{E}_t^2$ w.r.t $\boldsymbol{\theta}$:
10: $d\boldsymbol{\theta} \leftarrow d\boldsymbol{\theta} + \partial \mathcal{E}_t^2 / \partial \boldsymbol{\theta}$
11: Update weights $\boldsymbol{\theta}$ with $d\boldsymbol{\theta}/k$

C. GCNNs

A GCNN comprises stacked GC layers (GCLs). We use the GCNN method proposed by Schlichtkrull et al. [13] due to its computational efficiency and availability of directed graphs. Their method extracts features considering distant vertices in a graph by using multiple stacked layers rather than extracting features at once with a single one layer.

Here, a graph is represented by $G = (\mathcal{N}, E)$, where $\mathcal{N}$ is a set of vertices and E is a set of edges. The $l+1$-th layer output $H^{(l+1)}$ is calculated in a directed graph using the output of the l-th layer $H^{(l)}$ as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1}\tilde{A}H^{(l)}\Theta^{(l)}\right),$$

where each notation is defined as follows:

- $\tilde{A} \coloneqq A + I$ is the matrix added the identity matrix $I \in \mathbb{R}^{N\times N}$ to an adjacent matrix $A \in \mathbb{R}^{N\times N}$ of graph G, where N is the number of vertices in $\mathcal{N}$.
- $\tilde{D} \coloneqq D + I$ is the matrix added identity matrix I to a degree matrix $D \in \mathbb{R}^{N\times N}$ of graph G.
- $H^{(l)}$ is the l-th layer output, where $H^{(0)} = X \in \mathbb{R}^{N\times C}$ and X is a C-dimensional input feature vector per unit (e.g., the number of vehicles and average velocity).
- $h^{(l)}_{n,m_l} \in H^{(l)}$ is $n \in N$-th unit output of $m_l \in$ the $F^{(l)}$-th filter, where $F^{(0)} = C$.
- $\Theta^{(l)} \in \mathbb{R}^{F^{(l)}\times F^{(l+1)}}$ represents the neural network weights in the l-th layer, and $\theta^{(l)}_{m_l,m_{l+1}}$ represents the weights from the $m_l \in F^{(l)}$-th inputs to the m_{l+1}-th filter.
- $\sigma(\cdot)$ is an activation function, e.g., $\mathrm{ReLU}(\cdot) \coloneqq \max(0, \cdot)$.

To facilitate understanding, we show an example of each layer's output with a three-layered GCNN with a single filter in a 5×5 grid graph in Figure 2. We illustrate the connections of the second GCL units related to the third layer unit output in Figure 2(a). The other figures show examples of each layer's output. The areas surrounded by the dash lines indicate non-zero areas in the sub-figures when 1 is input to the center unit $h^{(0)}_{13,0}$ and 0 is input to the other units as input X. The results show that distant unit values propagate by stacking a greater number of layers. The GCNN has the smaller number of weights Θ than the FCNN because the GCNN connects a unit to the neighboring units of the next layer, while the FCNN connects a unit to all units. For example, in a 5×5 grid graph, the number of connections is four per unit in the GCNN (25 in the FCNN). This difference increases in both larger graphs and deeper neural networks, and helps to efficiently train neural networks using smaller datasets.

III. RL-BASED TRAFFIC SIGNAL CONTROL WITH GCNN

Here, we introduce a traffic signal control based on NFQI with a GCNN.

a) Cost Function: The objective of traffic signal control is to minimize the total travel time relative to all vehicles. However, the total travel time is unavailable as a cost because it cannot be calculated instantaneously at each time step. We use the following total wait time over all vehicles as the cost function r rather than the travel time.

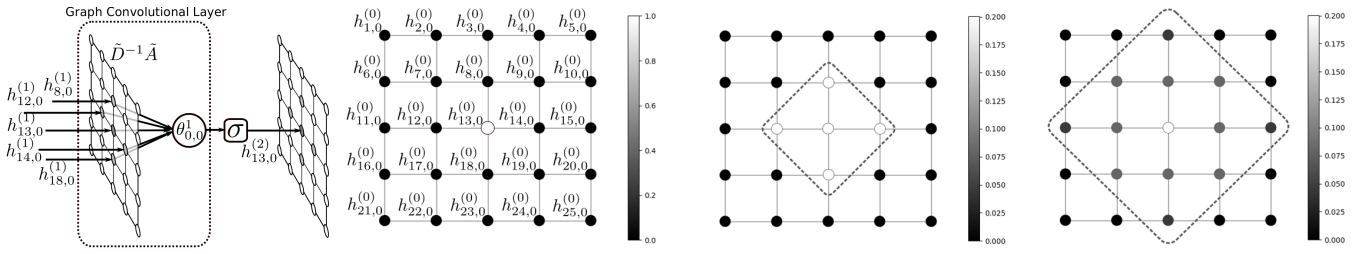
$$r_{i,t} \coloneqq \sum_{j\in\mathcal{V}_i} w_{j,t},$$
$$w_{j,t} \coloneqq \begin{cases} 0 & (v_{j,t} > 0.1) \\ w_{j,t-1} + \Delta t & (otherwise) \end{cases}, \qquad (2)$$

where suffix $i \in \mathcal{I}$ is an intersection index in a set of intersections $\mathcal{I}$. In addition, suffix t denotes time step, $\mathcal{V}_i$ is a set of vehicles on roads linked to an intersection i, and $w_{j,t}$ is the wait time of vehicle $j \in \mathcal{V}_i$ at time t. $v_{j,t}$ denotes the speed of vehicle j at time t, and $w_{j,t}$ is defined as the time spent by $v_{j,t}$ below 0.1 m/s since the last time it was faster than 0.1 m/s (Eq. (2)). In other words, the Q-value at intersection i is calculated as follows:

$$Q_i(s_t, a) = r_{i,t} + \mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a)}\left[\gamma V_i(s_{t+1})\right].$$

b) State and Action Representation: We use two feature variables per lane: vehicle queue length and average velocity. Here, a state is represented by a $2N$-dimensional feature vector, where N is the number of lanes. Note that the action space is a set of each lane's permission to enter intersections. We learn policies for each intersection in a distributed manner when more than one intersection is in a road network to avoid exponentially increasing the size of the action space.

c) NFQI with GCNN: We do not discuss the learning algorithm for NFQI with the GCNN in detail because the NFQI algorithm (Algorithm 1) can be used without modification. The weights of the GCNN are updated using backpropagation, and the number of GCNN weight vectors



(a) Graph convolutional layer (b) $H^{(0)}(=$ input $X)$ (c) $H^{(1)}$ (d) $H^{(2)}$

Fig. 2: Example of layer output with a three-layer graph GCNN in a 5×5 grid graph when inputting $1(= h^{(0)}_{13,0})$ to the center unit and 0 to other units as input X. (a) Connections of the second GGL units related to the output of the third layer's unit. (b)-(d) Examples of the output of each layer. The dash lines denote non-zero areas in the sub-figures. The results show that deeper layer output is propagated from more distant vertex values. The color bar represents the value scale.

is generally much smaller than the number of FCNN weights in traffic signal control domains because each lane connects significantly fewer lanes than the total number of lanes in a road network.

IV. Numerical Experiments

We evaluated the proposed method, NFQI with GCNN, using a traffic simulator by comparing a fixed-time control algorithm and NFQI with an FCNN.

A. Simulation Settings

We used the well-known open source traffic simulator, SUMO [16], to simulate traffic and traffic signal control. We evaluated the proposed method with six intersections connected to four six-lane roads, each shown in Figure 3. Here, each road length connected to lanes was set to 400 m, the road speed limit was 50 km/h, the vehicle length was 5 m, and the minimum gap between vehicles was 2.5 m. We simulated traffic for 500 s per trajectory.

The vehicles' origins and destinations were set to any lane endpoints in a uniformly random manner. A vehicle generation rate of 0.5, which indicates that 0.5 vehicles are generated per second in a road network, was used for learning, and vehicle generations rates between 0.25 and 4.0 vehicles per second were used for evaluation. In addition, each vehicle randomly selected a route from a set of shortest paths to proceed to their assigned destinations.

Traffic signal control actions at each intersection were represented by combinations of a traffic light controlling individual lanes for the entire intersection. We simplified the traffic signal patterns to reduce the action space because the total number of possible traffic light combinations grows exponentially relative to the number of roads and lanes connected to an intersection. We used the four patterns presented in Table I. For example, pattern #1 represents a case in which the traffic signals of all lanes on the road headed north and south and the rightmost lane of the roads headed east/west are green. Pattern #2 represents a case in which the traffic signals of the leftmost lane on the road headed north and south and the rightmost lane on the road headed east and west are green. Patterns #3 and #4 represent cases wherein north and south flips east and west. Here, the

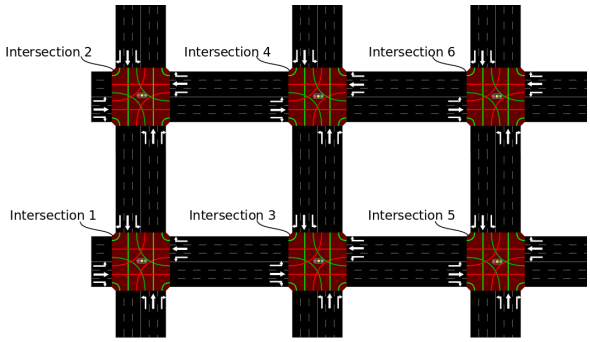


Fig. 3: Traffic signal control domain with six intersections connected to four six-lane roads. The objective is to minimize the cumulative total wait time of vehicles over all lanes. Each traffic signal is controlled by a separately learned policy.

duration of each phase was set to 30 s for patterns #1 and #3 and 6 s for patterns #2 and #4 when controlling traffic signals in a fixed-timing manner. In addition, the offset was set to 0 s, which represents the time lag between the start of each pattern for successive intersections.

TABLE I: Traffic signal patterns

	North and South			East and West			Fixed-timing control
	Right	Center	Left	R	C	L	Duration
#1	Green	Green	Green	Green	Red	Red	30 s
#2	R	R	G	G	R	R	6
#3	G	R	R	G	G	G	30
#4	G	R	R	R	R	G	6

B. Neural Network Structure

We employed a neural network with GCLs and a dueling network structure [21]. The dueling network structure improves the learning efficiency and stability by splitting the state action value function $Q(s, a)$ to the state value function $V(s)$ and the advantage function $A(s, a)$ as follows:

$$\hat{Q}(s, a; \boldsymbol{\theta}) = \hat{V}(s; \boldsymbol{\theta}) + \left(\hat{A}(s, a; \boldsymbol{\theta}) - \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \hat{A}(s, a; \boldsymbol{\theta}) \right),$$

where s, a, and $\boldsymbol{\theta}$ denote a state, an action, and the weights of the neural network, respectively.

We show the neural network structure used for our traffic signal control in Figure 4. Here, the input layer has 102 units (i.e., 17 roads $\times$6 lanes). The second and third layers are set as GCLs with 102 units. The fourth and fifth layers are forked to implicitly approximate $\hat{V}(s;\boldsymbol{\theta})$ and $\hat{A}(s,a;\boldsymbol{\theta})$, same as the dueling network. In addition, the fourth and fifth layers are FCLs with 51 and 25 units, respectively. In this experiment, the size of the fifth layer's outputs $\hat{V}$ and $\hat{A}$ was 1 and 4 each, and the size of the sixth layer's outputs $\hat{Q}$ is 4, which is the same number of traffic signal patterns. The softplus function and rectified liner unit were applied as an activation function for fifth layer output and ones for others respectively. Note that we used FCNN in which all GCLs in the above network were replaced with FCLs for comparison. All other settings (e.g., unit sizes) were the same as the network with the GCLs.

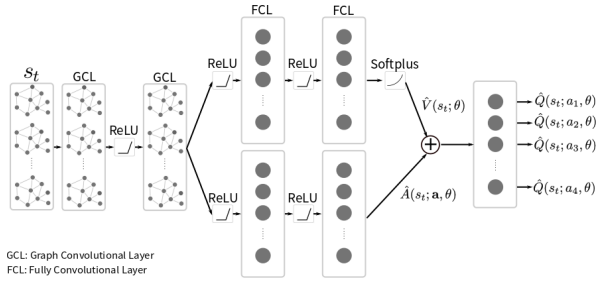


Fig. 4: Dueling network structures with GCNN

C. Training Dataset and Evaluation Metrics

We used traffic data controlled with fixed-timing traffic signals because we could collect appropriate real-word data easily. We collected 1,200 trajectories, where a vehicle generation rate is 0.5, i.e., 0.5 vehicles were uniformly generated at a road endpoint at random. Each trajectory involved a 500 s simulation.

We evaluated the policies according to two metrics, i.e., the mean (or median) wait time $\overline{W}_i$ at each intersection i over all vehicles and time steps and the mean of the mean (or median) wait time $\overline{W}$ over all intersections. The number M of conducting experiments for taking averages of the two metrics is set to $M = 20$. These metrics are calculated as follows:

$$\overline{W}_i = \begin{cases} \text{Mean}\left(\frac{1}{|\mathcal{V}_i|T}\sum_{t=0}^{T}\sum_{j\in\mathcal{V}_i} w_{m,j,t}\right) \\ \text{Median}\left(\frac{1}{|\mathcal{V}_i|T}\sum_{t=0}^{T}\sum_{j\in\mathcal{V}_i} w_{m,j,t}\right) \end{cases},$$

$$\overline{W} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\overline{W}_i,$$

where $\mathcal{V}_i$, $\mathcal{I}$, and $w_{m,j,t}$ are a set of vehicles on roads connected to intersection i, a set of intersections in the environment, and the waiting time of vehicle j at intersection i when m-the experiment is conducted, respectively. Two functions Mean$(\cdot)$ and Median$(\cdot)$ return a mean value and a median value over the experiments.

D. Results

We compared NFQI using the GCNN with NFQI using the FCNN and fixed-timing control in the six-intersection environment. The policies of each intersection were learned simultaneously but independently with different networks to avoid increasing the action space exponentially. In other words, a traffic signal at intersection i learned a policy to minimize the cumulative total wait time over all vehicles on roads connected to intersection i and the neural network weights for intersection i were not shared to the networks of other intersections.

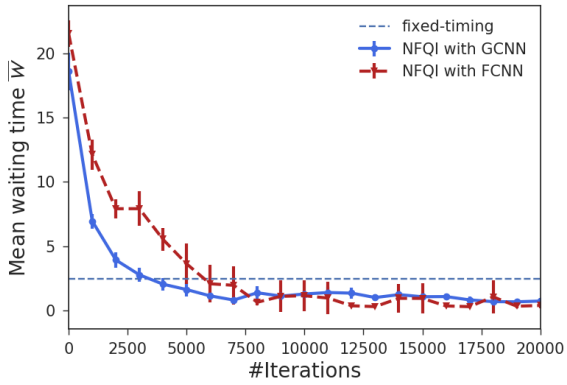


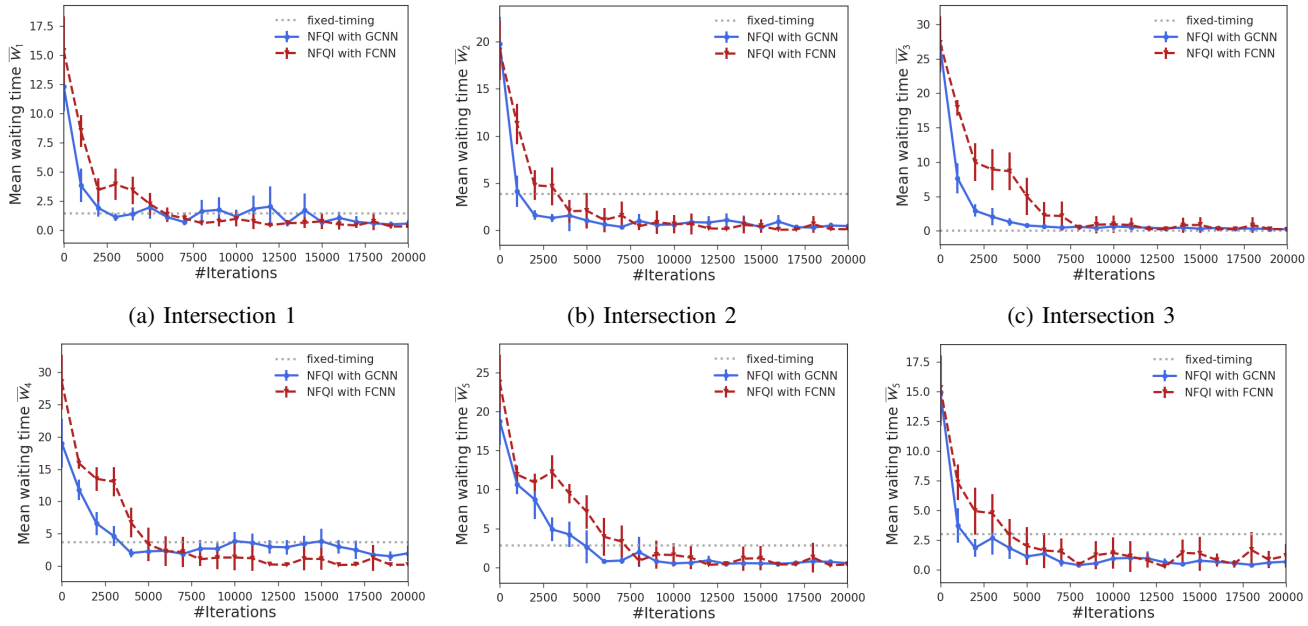
Fig. 5: Learning curve of mean wait time for NFQI with FCNN (red dash dots) and NFQI with GCNN (blue line) in the six-intersection domain. The mean wait time of fixed-timing control is indicated by the horizontal gray dotted line. The unit on y-axis is seconds. The error bars indicate a 95% confidence interval.

Figures 5 and 6 show the learning curves of the mean wait time $\overline{W}_i$ at each intersection i and the mean over all intersections $\overline{W}$ for the compared methods. The error bars in the figure indicate a 95% confidence interval. The results show that NFQIs with each network converge to policies better than those considered for fixed-timing control. NFQI with the GCNN converged to comparable policies twice as fast as that with the FCNN at the time achieved to comparable policies with fixed-timing control. These results would be reasonable because NFQI with the FCNN includes a GCNN as a neural network, but the size of the weight vector is much larger than that of NFQI with the GCNN. Table II lists the mean wait time and the median one at each intersection at 20,000th iterations. NFQI could find the policy with an approximately 2-s shorter wait time than fixed-timing control.

We tested the generalization performance by evaluating the compared methods with different vehicle generation rates during training. We learned policies with a vehicle generation rate of 0.5 and tested policies with rates between 0.25 and 4.0 vehicles per second. Figure 7 shows the mean of the mean wait time over all intersections for each vehicle generation rate. The error bars in the figure indicate a 95% confidence interval. The result shows that NFQIs maintain comparable

TABLE II: Average wait time at each intersection

		$\overline{W}$	$\overline{W}_1$	$\overline{W}_2$	$\overline{W}_3$	$\overline{W}_4$	$\overline{W}_5$	$\overline{W}_6$
Fixed timing	Median	2.5 s	1.4	3.8	0.0	3.7	2.8	3.0
	Mean	2.5	1.4	3.9	0.0	3.7	2.8	3.0
NFQI with FCNN	Median	0.43	0.31	0.77	0.25	0.30	0.67	0.28
	Mean	0.39	0.31	0.12	0.21	0.18	0.39	1.2
NFQI with GCNN	Median	0.27	0.14	0.42	0.29	0.13	0.26	0.38
	Mean	0.73	0.56	0.46	0.23	2.0	0.56	0.67



(a) Intersection 1 (b) Intersection 2 (c) Intersection 3

(d) Intersection 4 (e) Intersection 5 (f) Intersection 6

Fig. 6: Learning curves of mean wait time for NFQI with FCNN (red dashed line) and NFQI with GCNN (blue line) at each intersection. The mean wait time of fixed-timing control (no training process) is shown by the horizontal gray dotted line. The unit on y-axis is seconds. The error bars in the figure indicate a 95% confidence interval.

or shorter wait times than fixed-timing control in the entire range. The traffic signals controlled by NFQI maintained shorter wait times for a broader range of vehicle generation rates.

V. Concluding Remarks

In this study, we developed a traffic signal control method based on batch RL with a GCNN. The proposed method learns a policy based on NFQI with traffic features extracted for peripheral roads in a road network with multiple intersections using stacked GCLs without manually designed features.

We numerically evaluated the proposed method using a traffic simulator in a six-intersection environment. We implemented each agent in a distributed manner to learn a policy to control a traffic signal at an intersection using the proposed method rather than communicating with other agents. The results showed that NFQI with the GCNN obtained comparable policies twice as fast as NFQI with an FCNN. We also tested the generalization performance of our method by evaluating the learned policies using different

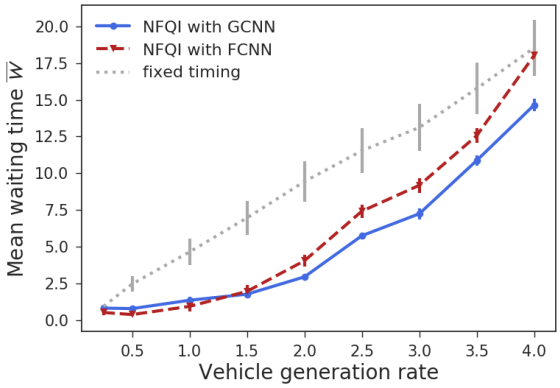


Fig. 7: Vehicle generation rate vs. mean wait time for NFQI with FCNN (red dash dots), NFQI with GCNN (blue line), and fixed-timing control (gray dots). The error bars in the figure indicate a 95% confidence interval.

vehicle generation rates. The result showed that policies learned using the proposed method could maintain control with the shortest mean wait times for a broader range of traffic demand changes.

In the future, we plan to apply the proposed method to real-world traffic data. Furthermore, we plan to further improve the waiting time using a recently proposed RL method, A3C [22], and extend the proposed method to control multi-modal traffic flows, e.g., flows that include vehicles, pedestrians, and bicyclists.

REFERENCES

[1] Dongbin Zhao, Yujie Dai, and Zhen Zhang, "Computational Intelligence in Urban Traffic Signal Control: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 485–494, jul 2012.

[2] M. Alsabaan, W. Alasmary, A. Albasir, and K. Naik, "Vehicular Networks for a Greener Environment: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1372–1388, 2013.

[3] B. P. Gokulan and D. Srinivasan, "Distributed Geometric Fuzzy Multiagent Urban Traffic Signal Control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 714–727, sep 2010.

[4] J. García-Nieto, E. Alba, and A. Carolina Olivera, "Swarm intelligence for traffic light scheduling: Application to real urban areas," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 2, pp. 274–283, mar 2012.

[5] D. Srinivasan, M. Choy, and R. Cheu, "Neural Networks for Real-Time Traffic Signal Control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 261–272, sep 2006.

[6] P. LA and S. Bhatnagar, "Reinforcement Learning With Function Approximation for Traffic Signal Control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412–421, jun 2011.

[7] P. Mannion, J. Duggan, and E. Howley, "An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control," in *Autonomic Road Transport Support Systems*. Cham: Springer International Publishing, 2016, pp. 47–66.

[8] Prashanth L. A. and Shalabh Bhatnagar, "Reinforcement Learning With Function Approximation for Traffic Signal Control," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 412–421, 2011.

[9] W. Genders and S. Razavi, "Using a Deep Reinforcement Learning Agent for Traffic Signal Control," *arXiv*, 2016.

[10] 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," *arXiv*, pp. 1–10, may 2017.

[11] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional Networks on Graphs for Learning Molecular Fingerprints."

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering."

[13] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling Relational Data with Graph Convolutional Networks," *arXiv*, pp. 1–12, 2017.

[14] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *ICLR*, 2017.

[15] M. Riedmiller, "Neural fitted Q iteration - First experiences with a data efficient neural Reinforcement Learning method," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2005.

[16] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "SUMO (Simulation of Urban MObility) An open-source traffic simulation CarDriver Model," *Proc of the 4th Middle East Symposium on Simulation and Modelling*, pp. 183–187, 2002.

[17] J. Tsitsiklis and B. van Roy, "Analysis of Temporal-Diffference Learning with Function Approximation," *Advances in Neural Information Processing Systems 9 (NIPS)*, vol. 42, no. 1988, pp. 1075–1081, 1997.

[18] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-Based Batch Mode Reinforcement Learning," *Journal of Machine Learning Research*, vol. 6, pp. 503–556, 2005.

[19] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," pp. 283–290, 1996.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[21] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," no. 9, 2015.

[22] V. Mnih, A. Apuigdo Enech Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning."