Applied Soft Computing 119 (2022) 108497

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Area-wide traffic signal control based on a deep graph Q-Network (DGQN) trained in an asynchronous manner

Gyeongjun Kim, Keemin Sohn *

Department of Urban Engineering, Chung-Ang University, 84 Heukseok-ro, Dongjak-gu, Seoul 156-756, Republic of Korea

ARTICLE INFO

Article history:
Received 25 September 2020
Received in revised form 15 December 2021
Accepted 17 January 2022
Available online 29 January 2022

Keywords:
Adaptive traffic signal control
Deep graph Q-network (DGQN)
Graph convolution
Reinforcement learning

ABSTRACT

Value-based reinforcement learning (RL) algorithms have been widely applied in traffic signal studies. There are, however, several problems in jointly controlling traffic lights for a large transportation network. First, the discrete action space exponentially explodes as the number of intersections to be jointly controlled increases. With its model structure, the original deep Q-network (DQN) could not accommodate a large action space. The problem was resolved by revising the output structure of a DQN holding the framework of a single-agent RL algorithm Second, when mapping traffic states into an action value, it is difficult to consider spatio-temporal correlations over a large transportation network. A deep graph Q-network (DGQN) was devised to efficiently accommodate spatio-temporal dependencies on a large scale. Finally, training the proposed DGQN with a large number of joint actions requires much time to converge. An asynchronous update methodology with multiple actor learners was devised for a DGQN to quickly reach an optimal policy. By combining these three remedies, a DGQN succeeded in jointly controlling the traffic lights in a large transportation network in Seoul. This approach outperformed other "state-of-the-art" RL algorithms as well as an actual fixed-signal operation. The proposed DGQN decreased the average delay of the current fixed operation to 55.7%, whereas those of reference models DQN-OGCN and DQN-FC were 72.5 and 92.0%, respectively.

1. Introduction

Many researchers have rigorously studied the application of value-based reinforcement learning (RL) to traffic light control. When network-wide traffic signals are jointly controlled, a multi-agent RL model prevails because of the increasing size of state and action spaces. In a multi-agent RL setting for jointly controlling traffic lights, agents cooperate to achieve a common goal such as the minimization of total delay. This algorithm, however, cannot guarantee a global optimum without full coordination with other agents' actions. More concretely, it cannot break ties between agents in such a cooperative environment [1]. For example, when jointly controlling two consecutive intersections, at a moment the phase combination allowing the north–south through traffic for both intersections may have the same action-value as the phase combination allowing the north–south left-turn traffic for both intersections. If there is no coordination between upstream and downstream controllers, the downstream controller may choose the through-phase expecting the upstream controller to select the same phase, although the upstream controller could actually select the left-turn phase expecting the downstream controller to do the same. This leads to a non-optimal pair of signal phases. The lack of coordination thus prevents a multi-agent RL algorithm from reaching a global optimum. A single-agent DQN could solve this problem, but the model has a difficulty in accommodating a large action space.

Our previous study revised the output structure of the original deep Q-network (DQN) to accommodate a large action space within the framework of a single-agent model [2]. Whereas the extended model had been validated for a small-scale synthesized network composed of only 4 intersections, in the present study the model was applied to jointly control a real transportation network containing 15 intersections. When jointly controlling the traffic signals of 15 different intersections, the original DQN requires the total number of output nodes to be no less than 4^{15} when each intersection has 4 available signal phases. The number of parameters to be learned also explodes, and an unrealistic amount of computing resources is necessary. On the other hand, the extended DQN requires no more than 4×15 output nodes while accommodating the full coordination between joint traffic signals of individual intersections. This scheme also offers a great advantage for training the model by facilitating the selection of a joint action in the Q-learning algorithm without the full enumeration of joint actions (see Section 3). The architecture of the extended DQN was described in a rigorous manner using mathematical expressions.

* Corresponding author.
E-mail addresses: kkjunn7033@cau.ac.kr (G. Kim), kmsohn@cau.ac.kr (K. Sohn).

https://doi.org/10.1016/j.asoc.2022.108497
1568-4946/

The second problem is posed when a RL-based traffic signal control is implemented in a complex transportation network. It is inefficient for either a fully connected (FC) layer or a 2-D convolutional layer to recognize the topology of a transportation network. Thus, those methods require an inordinate amount of computing time to accommodate the spatial dependencies between the traffic states of different intersection approaches. A transportation network has a constant topology through which traffic states propagate. If a graph convolution is employed in the structure of a Q-network, the dependency of the traffic state of a given intersection on the traffic states of other intersections can be efficiently addressed when approximating the true action values. Some researchers have pioneered the use of graph convolutions to constitute a Q-network for traffic signal control [3,4]. Their graph convolutions, however, depended on a constant adjacency matrix, and their test bed included only a few intersections and depended on a multi-agent RL algorithm that could not fully coordinate the signal phases of different intersections. We devised a novel graph convolution scheme that can vary the connection intensities by time lags and incorporate them into an extended DQN for large-scale joint traffic signal control. With this graph convolution scheme, a deep graph Q-network (DGQN) was developed to accommodate the spatio-temporal dependencies in a transportation network. The DGQN made it possible to accommodate the traffic states of upstream and downstream roads from past times in order to more efficiently determine the present traffic signals of an intersection approach. It is also possible to use either a FC or a 2-D convolutional layer to recognize spatial correlations. A DGQN, however, is the most efficient model because it directly uses the topology of a road network to derive action values from traffic states.

The difficulty in training a DGQN to jointly control traffic lights on a large scale was the last issue to be addressed in the present study. Even though the proposed DGQN can deal with a large number of actions without an exorbitant use of computer memory, the large action space remains unchanged and should be explored sufficiently during training. For convergence, a DGQN must experience as many transitions as possible from an action-state pair to a subsequent state. In this context, a single environment is not sufficient for a RL algorithm to reach the convergence within a practical computation time. In the present study, multiple environments were set up and asynchronous updates were applied in training the DGQN. Unlike the existing A3C algorithm developed by Mnih et al. [5], a replay memory was separately kept for each actor-learner. Four simulated environments were used, and each environment had a different level of traffic demand. By applying the asynchronous training scheme, the DGQN can deal with a large number of joint actions for area-wide traffic light control.

The proposed DGQN essentially took the form of a single-agent RL model rather than the multi-agent RL models that prevail in the field of study for joint traffic signal control at a network level. Nonetheless, combining the three remedies outlined above made it possible to find an optimal policy that could jointly control the traffic signals of 15 intersections in a real transportation network. The DGQN outperformed a DQN with only FC layers and an original graph convolutional neural network (GCN) with a constant adjacency matrix. Furthermore, an asynchronous learning scheme made it possible for a DGQN to converge to an optimal policy within a practical amount of computing time.

The next section introduces a literature search that dealt with the applications of a RL algorithm to traffic signal control. In the third section, the state, action, and reward are defined for the proposed DGQN. The fourth section describes the structure of the proposed DGQN to circumvent the exorbitant memory usage with respect to a large action space. An asynchronous training algorithm is devised in the fifth section. How to set up simulation experiments is described in the sixth section. The seventh section shows analysis results and performance comparisons with other reference models. In the last section, conclusions are drawn, and further studies are discussed.

2. Related work

Studies that employed a RL-based algorithm to jointly control traffic signals in a transportation network are investigated in this section. Prior to the advance of deep learning technologies, model-based multi-agent RL algorithms had already been adopted in joint traffic signal control to gain an advantage of saving computing resources in a distributed manner [6–11]. Wiering [6] established an RL model prototype for traffic signal control based on multiple agents. Later, Kuyer et al. [8] extended Wiering's model by employing a Max-Plus algorithm to pass messages among neighboring agents, which facilitates coordination between actions of different agents. Houli et al. [9] devised a vehicular ad hoc network within the framework of a multi-agent RL to streamline the information exchange among agents. More practically, a partially coordinated multi-agent RL algorithm was developed by El-Tantawy et al. [11], which can be applied to real-world applications by allowing each agent to communicate only with its first-order neighbors and to partition the entire state space into subspaces consisting of a pair of agents. All these early studies, however, adopted a tabular update of the Q-function and did not approximate their Q-function using a deep neural network.

A model-free RL algorithm that approximates the Q-function using a deep neural network started to emerge for traffic light control after a DQN had succeeded in the playing of classic video games [12]. There are quite a few references in the literature to the adoption of a deep-learning model to address issues associated with joint traffic signal control in a large transportation network. Some researchers used a DQN within a multi-agent RL framework to solve the joint traffic signal control problem [13–16]. However, they did not ensure that an exact solution was found because only partial coordination between agents was considered. Van der Pol and Oliehoek [16] decomposed the global Q-function into local Q-functions and used a Max-Plus algorithm to exchange messages between agents within a DQN framework. Tan et al. [15] used a hierarchy to decompose a RL problem, and a global agent utilized the achievements from local agents to perform its own action.

Some pioneers have begun to use a graph-based neural network to approximate the true Q-function when traffic lights are jointly controlled in a transportation network. When constructing a DQN, Nishi et al. [4] used graph convolutional layers to accommodate spatial dependencies between traffic states separated geographically in a transportation network. An adjacency matrix that indicates the first-order connections in a road network was prepared in advance. In their framework, however, each traffic signal is controlled by a separately learned policy, which means they could not find a global joint policy. More recently, Wang et al. [3] adopted an attention-based graph convolution to consider spatial dependencies between traffic states when approximating the true Q-function. The directional adjacency graph of traffic lights was explicitly constructed for modeling the geographical structure information to facilitate coordination among multi-intersection traffic light control. A recurrent neural network was also integrated with the graph units to accommodate temporal dependencies. Spatio-temporal correlations between traffic states were recognized simultaneously within a DQN framework. Their DQN was evaluated simply in a distributed manner whereby the global Q-function is decomposed into local

independent Q-functions. Basically, a multi-agent RL algorithm cannot obtain a globally optimal policy, unless an agent's action can accommodate full coordination with the actions of other agents [1].

Besides the value-based RL algorithms, a policy-gradient RL algorithm was also adopted to solve joint traffic signal control in a large transportation network [17–20]. A policy-based RL algorithm is applicable for continuous control problems to determine the traffic signal phase duration with the phase sequence fixed, whereas a value-based RL algorithm optimizes the phase sequence for constant phase duration. The policy-based RL algorithm employs two deep neural networks, each of which approximates an action value function and a policy function, respectively. The control of the algorithm is known to be more consistent than that of a value-based RL algorithm [21,22]. Casas [19] applied a single-agent actor–critic algorithm for the traffic signal control of multiple intersections, and Chu et al. [20] employed a multi-agent actor–critic algorithm that shares partial information between agents to jointly control multiple traffic lights. The latter study considered information of neighborhood policies to improve the observability of each local agent and introduced a spatial discount factor to weaken the state and reward signals from remote agents.

Fuzzy systems have also been used to optimize traffic signal controls [23,24]. Traffic light controllers based on a fuzzy system optimize the extension of phase duration with the phase sequence fixed, which is similar to an actor–critic RL algorithm in that both have a continuous action space. Some researchers have employed the neuro fuzzy systems for traffic signal control [25,26] wherein a neural network was designed to mimic the functions of a fuzzy system such as fuzzification, rule-based inference, and defuzzification. A fuzzy neural network can be trained on real or simulated data, whereas a naïve fuzzy system depends on human intuitions to set up fuzzy sets and rules. Although this neuro fuzzy technology provides a plausible way to integrate rule-based and learning-based models, the fuzzy traffic light controller is yielding its position to a RL algorithm embedded with deep neural networks. Recently, Kumar et al. [27] attempted to incorporate the fuzzy inference with a RL algorithm, so that the former selected a specific mode and the latter controlled traffic lights according to the chosen mode. The proposed DGQN was compared with the two different types of algorithms introduced above. The comparison results will be discussed in Sections 7.2 and 7.3.

3. The definitions of state, action, and reward

3.1. The definition of state

Conventional traffic parameters such as delays and queue lengths are a decisive measure to evaluate the performance of an adaptive traffic signal control system. In the field of study for RL-based traffic signal controls, the traffic delay and queue length have widely been chosen to represent the traffic state [11,28–32]. The present study also adopted both of these parameters as the state of the traffic environment. However, there may be skepticism regarding how to accurately measure these parameters when an RL algorithm is implemented in the field. Our two previous studies showed strong evidence that they can be measured onsite based solely on video images [33,34].

In this context, the traffic delay and queue length were selected as the state of the present RL algorithm. The state was defined for each lane group that is comprised of one or more lanes that share a common stop-line and capacity. Generally, all exclusive turn lanes are treated as separate lane groups. Through lanes are also generally grouped together, including through lanes that allow for shared right and/or left-turn movements. Feasible phases and lane groups were recognized for each intersection in the testbed prior to establishing a DQN model.

3.2. The definition of action

There are two types of actions in a RL-based traffic signal control problem. A continuous action allows a phase duration or its proportion to a cycle length to be determined [19,35–37]. In this case, the displaying sequence of given phases had to be fixed. On the other hand, for a discrete action any feasible phase can be selected for a time period [6,11,30]. For the discrete action, the duration of a signal phase cannot be shortened below the predefined interval but should be a multiple value of the interval. The present study chose the second option for the DGQN to secure a larger degree of freedom in a traffic signal control. Feasible phases for each intersection were selected relative to the real operation of a traffic signal control in the testbed. For the discrete action space, it should be noted that the number of joint actions explodes exponentially with increases in the number of intersections to be jointly controlled.

For comparison, the present study offered an alternative setting of continuous action to three reference models. An actor–critic RL algorithm was chosen wherein the duration of each traffic signal phase was the determinant variable with the phase order predefined. A conventional fuzzy system that returns a crisp control variable was tested. A fuzzy neural network was also chosen for comparison. For both fuzzy models, each phase duration was determined between predefined minimum and maximum green times.

3.3. The definition of reward

The goal of joint traffic signal control is to minimize the total delay at the entire transportation network level. The conventional algorithm for adaptive traffic light control has not achieved this goal in the real world. Most adaptive signal control systems cannot help leading to a local optimum. For example, applying the best offset for a major corridor may undermine minor traffic flows, which is frequently observed in the field. Even though a multi-agent RL algorithm chooses a delay-dependent reward specification, it ends up with a local solution since it should be learned in a distributed manner without the full action coordination. The present study uses a single reward that is derived from the total cumulative delay in an entire transportation network. A single agent finds an optimal traffic signal policy so that the total system delay can be minimized. At the end of every learning interval, the reward is set as $+1$ if the delay cumulated during the current interval is less than that cumulated during the previous interval and is set as -1 otherwise.

4. The structure of DGQN

4.1. An overview of graph convolution

A 2-D convolution cannot efficiently extract correlations between nodes in a graph. Kipf and Welling [38] simplified a graph convolution method by confining the filters' operation to only the first-order neighbors. Repeating the graph convolution, however, extends to remote nodes from a target node. The working principle of the graph convolution is as follows.

Given a graph, $G = (V, E)$, a GCN requires two matrices as input. A feature matrix ($X \in \mathbb{R}^{N \times F^0}$) and an adjacency matrix ($A \in \mathbb{R}^{N \times N}$) is fed into a GCN. A hidden layer of the GCN is generated by a propagation rule based on the previous hidden

layer and an adjacency matrix that represent a graph topology . How to choose the propagation rule is the key in the concept of GCN. Eq. (1) is the conventional form of a layer-wise propagation rule, wherein weights are separated from a constant adjacency matrix.

$$H^l = \sigma\left(AH^{l-1}Q^{l-1}\right) \tag{1}$$

N	= number of vertices
F^0	= number of given features for a node
F^l	= number of neurons (dimension) of l^{th} hidden layer
V	= set of vertices (=nodes)
E	= set of edges
H^0	= initial hidden layer (=X)
H^l	= l^{th} hidden layer
$\sigma(\cdot)$	= activation function such as a sigmoid and a rectified linear unit (ReLU)
$Q^{l-1} \in \mathbb{R}^{F^{l-1}\times F^l}$	= weight parameter matrix

However, there are two typical problems regarding Eq. (1). First, multiplying an adjacency matrix with a hidden-feature matrix cannot transfer a node's own features to the next layer. Only features of its neighbor nodes can be passed to the next layer. To circumvent this drawback, an identity matrix is added to the given adjacency matrix ($\tilde{A} = I + A$). Second, feature values scale up as the layer-by-layer propagation repeats. The average normalization of the adjacency matrix ($\tilde{A}$) can be an option to sidestep this problem. A diagonal node-degree matrix (D) is prepared such that its diagonal element sums up the row values of the adjacency matrix ($D_{ii} = \sum_j \tilde{A}_{ij}$). The adjacency matrix is then normalized by multiplying the inverse of the diagonal matrix and the given adjacency matrix ($D^{-1}\tilde{A}$), and thus all rows of the resultant matrix sum to one. On the other hand, instead of this average normalization, Kipf and Welling [38] adopted a spectral normalization to secure a greater dynamic for the propagation rule ($D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}$). The spectral normalization scheme divided an element of the adjacency matrix by the row sum and the column sum ($\frac{\tilde{A}_{ij}}{\sqrt{D_{ii}}\sqrt{D_{jj}}}$), whereas the average normalization scheme divided an element only by the row sum in ($\frac{\tilde{A}_{ij}}{D_{ii}}$). The final propagation rule established by Kipf and Welling [38] takes the form of Eq. (2).

$$H^l = \sigma\mathrm{e}\left(D^{-\frac{1}{2}}\tilde{A}D^{-\frac{1}{2}}H^{l-1}Q^{l-1}\right) \tag{2}$$

The concept of this graph convolution, however, is not appropriate for a case where the intensity of connection varies by adjacent edges. The propagation of traffic states is a typical example wherein the influence of a road segment's traffic state is exerted differently on its first-order neighbors. A fixed adjacency matrix cannot accommodate such traffic propagation. Recently, some researchers utilized graph-attention networks to assign different weights to neighboring nodes and to learn the weights from data [39–41]. Their model, however, is redundant when applied to a fixed-graph topology since it should train parameters for all possible connections between nodes. This type of a full-attention mechanism is inefficient for a problem involving a fixed graph such as a transportation network.

In our previous study [42], we devised a novel graph convolution scheme for a constant transportation network to overcome the redundancy and forecasted traffic speeds on an area-wide scale. This scheme was adapted to compose a DGQN for RL-based traffic signal control. The incorporation of a network topology into a Q-function facilitates the approximation of true action values from traffic states. The present study recomposed a transportation network such that vertices and edges represented lane groups and the connections between them, respectively. A variable adjacency matrix was prepared such that the row and column dimension of the adjacency matrix was set equal to the total number of lane groups in the testbed. For the row of a specific lane group, a non-zero value was assigned only for columns corresponding to first-order downstream and upstream neighbors (see Fig. 1). By doing so, different traffic propagation patterns were accommodated. That is, the traffic state of a lane group can be differently affected by upstream or downstream lane groups. Eq. (3) denotes how to parameterize the variable adjacency in an element-wise manner.

$$A^l_{ij} = \mathrm{softmax}\left(\theta^l_{ij}\right) = \frac{e^{\theta^l_{ij}}}{\sum_{k\in\mathfrak{N}_i} e^{\theta^l_{ik}}} \tag{3}$$

A^l	= variable adjacent matrix for lth hidden layer
A^l_{ij}	= ij element of A^l
θ^l_{ij}	= weight parameter in A^l
$\mathcal{R}_i$	= set of lane groups that are directly connected to the lane group i

Repeating a graph convolution makes it possible to recognize the influence of lane groups in remote intersections. For each graph convolution in a DGQN, a different adjacency matrix was used with the position of non-zero cells maintained, so that the influence of neighboring lane groups could be differentiated by spatio-temporal domain. The proposed graph convolution is different from the original one in that connection intensities within an adjacency matrix are not fixed but parameterized.

4.2. The architecture of a DGQN

4.2.1. The former portion of a DGQN to elicit features from states

The former portion of the proposed DGQN was designed to extract features from input states. The portion was built on novel graph convolutions with multiple adjacency matrices to reflect the different spatio-temporal effects of neighboring lane groups by connection order. As far as we could ascertain, the present study is the first traffic signal control attempt to use a Q-network based on graph convolutions with parameterized adjacency matrices.

Fig. 2 depicts the recursive expression of graph convolutions that constitutes a former portion of the DGQN. The row dimension for input (N) was set equal to the number of lane groups in the testbed, and the column dimension (P) of the input was set equal to the number of state variables. The number of graph convolutions was varied to consider the different spatio-temporal influences of input states. Four-dimensional weight parameters (θ^{kt}_{ij}) were included to differentiate the impact of input states for different time intervals. A tensor ($\boldsymbol{S}_t = [s_{t-2}\, s_{t-1}\, s_t]$) that concatenated the vectors of three previous time intervals was chosen as the RL state at time t, and each input vector was separately fed to subsequent graph convolutions so that an older traffic state could be processed through a greater number of graph convolutions. Since the proposed model did not include separate weight matrices (Q^l), a dimension for hidden layers was maintained through all graph convolutions.

Eqs. (4a) to (4d) represent graph convolution layers to accommodate the spatio-temporal influence of input states on action values. The last hidden tensor after graph convolutions goes through a 2-D convolution, and the output tensor is flattened and fully connected to a dense layer. The dense layer acts as an input to the latter portion of the DGQN.

$$H_{t-2} = \sigma\left(A^{32}_\theta \sigma\left(A^{22}_\theta \sigma\left(A^{12}_\theta s_{t-2}\right)\right)\right) \tag{4a}$$

$$H_{t-1} = \sigma\left(A^{21}_\theta \sigma\left(A^{11}_\theta s_{t-1}\right)\right) \tag{4b}$$

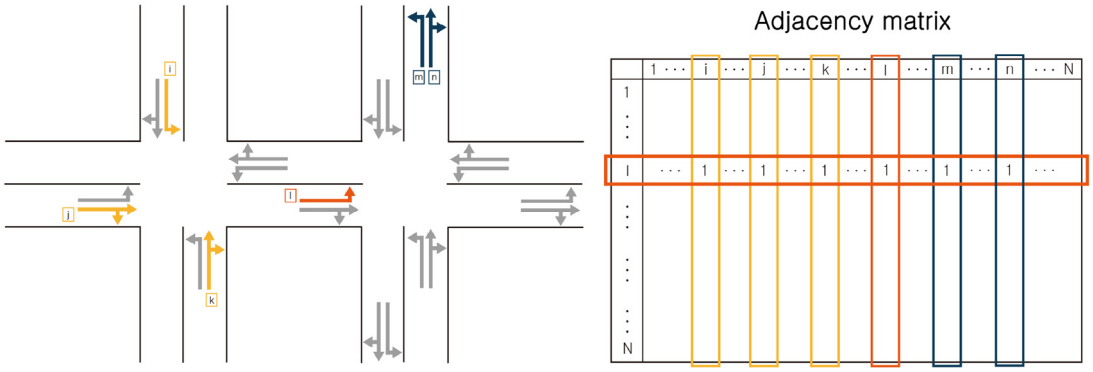


Fig. 1. The concept of adjacency matrix. The fixed adjacency matrix (right) is used as a mask to constitute the parameterized adjacency matrices in a DGQN.

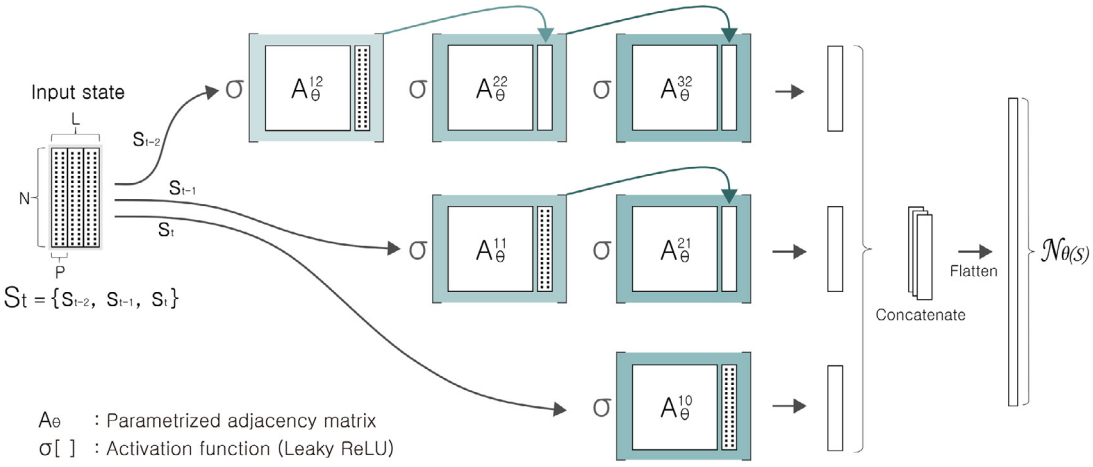


Fig. 2. The former parts of the proposed DGQN for graph convolutions.

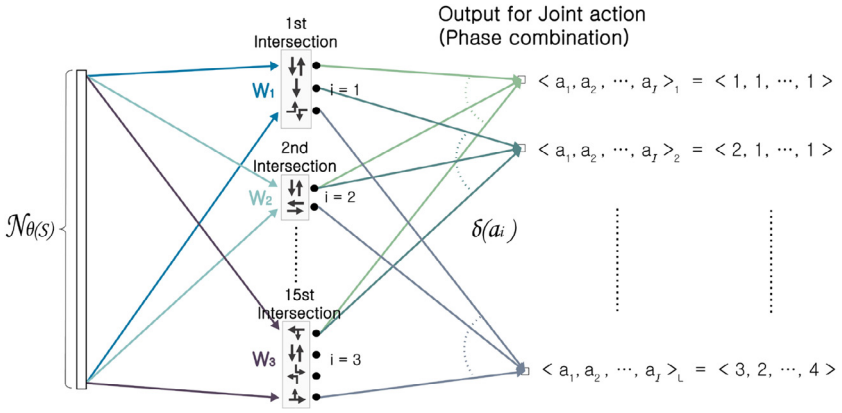


Fig. 3. The latter parts of the proposed DGQN to accommodate a larger action space.

$$H_t = \sigma\left(A_\theta^{10} s_t\right) \tag{4c}$$

$$\boldsymbol{H}_t = [H_{t-2}\ H_{t-1}\ H_t] \text{ for } t = 2, \ldots, T-2 \tag{4d}$$

s_t = input state vector in interval t

A_θ^{kl} = parameterized adjacency matrix to accommodate the k^{th} graph convolution for s_{t-l}

$\{\theta_{ij}^{kl}\}$ = parameter set included in A_θ^{kl}

H_t = hidden layer from s_t after going through graph convolutions

$\boldsymbol{H}_t$ = tensor that concatenates H_{t-2}, H_{t-1}, and H_t along the last axis

4.2.2. The latter portion of the DGQN to accommodate a large action space

The latter portion of the proposed DGQN maps the abstracted features into action values (see Fig. 3). A conventional DQN takes a specific form whereby the number of output nodes should be the same as the size of the action space. Thus, if the number of joint actions increases, the specification cannot work properly. Our previous study overcame this problem by selectively fixing weight parameters between the last hidden layer and the output layer [2]. This measure was mathematically expressed and embedded into the proposed DGQN. For brevity, the subscript for time index will be dropped from all notations hereafter. Eq. (5) represents the global Q-function that can be decomposed into

individual intersections with correlations between signal phases considered.

$$
\begin{aligned}
&Q\left(\boldsymbol{S}, \langle a_1, \ldots, a_I\rangle | \boldsymbol{\theta}, \langle \boldsymbol{W}_1, \ldots, \boldsymbol{W}_I\rangle\right) \\
&= \sum_{i=1}^{I} \sum_{j=1}^{\Phi} \mathcal{N}_{\boldsymbol{\theta}}^{T}\left(\boldsymbol{S}\right) \mathrm{w}_j^i \delta_j\left(a_i\right) \\
&= \sum_{i=1}^{I} \mathcal{N}_{\boldsymbol{\theta}}^{T}\left(\boldsymbol{S}\right) \boldsymbol{W}_i \boldsymbol{\delta}\left(a_i\right) \qquad (5)
\end{aligned}
$$

I = number of intersections to be jointly controlled

Φ = max. number of feasible signal phases for an intersection

N = number of lane groups (=77 for the present testbed)

P = number of attributes to represent a state (=2 for the present study including delay and queue length)

L = number of time lags (=3 for the present study)

$\boldsymbol{S}$ = $[N \times P \times L]$ state tensor representing the traffic states of the entire transportation network during the present and $(L-1)$ previous intervals.

$\langle a_1, \ldots, a_I\rangle$ = a joint signal phase (=action) with each $a_i \in \{1, \ldots, \Phi\}$ representing a specific signal phase for the i^{th} intersection

M = number of nodes for the last FC layer

$\mathcal{N}_{\boldsymbol{\theta}}\left(\boldsymbol{S}\right)$ = $[M \times 1]$ tensor corresponding to the last FC layer of the former portion of the DGQN

$\mathcal{N}_{\boldsymbol{\theta}}^{\boldsymbol{T}}\left(\boldsymbol{S}\right)$ = the transposed $\mathcal{N}_{\boldsymbol{\theta}}\left(\boldsymbol{S}\right)$

$\boldsymbol{W}_i$ = $\left\{\boldsymbol{w}_{kj}^i\right\}$, $[M \times \Phi]$ weight matrix that includes connections between the last FC layer and the de-facto output nodes for the i^{th} intersection, $k = 1, \ldots, M, j = 1, \ldots, \Phi$ and $i = 1, \ldots, I$

w_j^i = $\left[w_{1j}^i, \ldots, w_{Mj}^i\right]^T$, $[M \times 1]$ vector denoting the j^{th} column of the matrix $\boldsymbol{W}_i, j = 1, \ldots, \Phi$ and $i = 1, \ldots, I$

$\boldsymbol{\delta}\left(a_i\right)$ = $\left[\delta_1\left(a_i\right), \ldots, \delta_\Phi\left(a_i\right)\right]^T$, $[\Phi \times 1]$ indicator of the vector used to choose a phase given to the i^{th}

$$
\delta_j\left(a_i\right) = \begin{cases} 1 & \text{If } a_i = \text{the } j^{th} \text{ signal phase} \\ & \text{of the intersection i} \\ 0 & \text{Otherwise} \end{cases}, j = 1, \ldots, \Phi \text{ and } i = 1, \ldots, I
$$

In Eq. (5), $\mathcal{N}_{\boldsymbol{\theta}}\left(\boldsymbol{S}\right)$ is last hidden layer of the former portion of the DGQN following the graph convolutions. This layer can be interpreted as a function of an input traffic state ($\boldsymbol{S}$), given that $\boldsymbol{\theta}$ is a set of parameters $\{\theta_{ij}^{kl}\}$ associated with former graph convolutions to extract features from the traffic state. Elements of the indicator vector $[\boldsymbol{\delta}\left(a_i\right)]$ should meet the following conditions to guarantee that only a single signal phase is assigned to each intersection.

$$
\sum_{j=1}^{\Phi} \delta_j\left(a_i\right) = 1, i = 1, \ldots, I \qquad (6)
$$

The loss function represents the discrepancy between the target and the incumbent Q-network in terms of the Bellman optimality equation. In Eq. (7), $\boldsymbol{\theta}^-$ and $\langle \boldsymbol{W}_1^-, \ldots, \boldsymbol{W}_1^-\rangle$ are parameters of the target Q-network that are fixed when updating the parameters of the incumbent Q-function. To get the target parameters, the parameters of the incumbent Q-function are periodically copied on a long-term basis while learning. A replay set was maintained so that prior experiences $\{(\boldsymbol{S}, \langle a_1, \ldots, a_I\rangle, \boldsymbol{S}', r)\}$ could be stored. The loss function was minimized for each batch of examples taken randomly from the replay set. The loss function was reduced to the last term of Eq. (7) by substituting Eq. (5) for Q-functions and introducing a batch scheme instead of general expectations. In Eq. (7), B represents the batch size, and $\left(\boldsymbol{S}_b, \langle a_{1b}, \ldots, a_{Ib}\rangle, \boldsymbol{S}_b', r_b\right)$ is the b^{th} example in the batch. It is a great advantage that the maximum of the target Q-function can be obtained only by adding the largest node values in the de-facto output layer across all intersections. If the original DQN were used, a full enumeration of joint actions would be necessary for every iteration to solve the maximization problem. For convenience, the former and later portions of the network are separately drawn in Figs. 2 and 3. The first hidden layer of Fig. 3 is the same as the last FC layer of Fig. 2. The target Q-network also uses the same architecture but has a different set of parameters.

$$
\begin{aligned}
&\mathcal{L}\left(\boldsymbol{\theta}, \langle \boldsymbol{W}_1, \ldots, \boldsymbol{W}_I\rangle\right) \\
&= E_{\left(S, \langle a_1, \ldots, a_I\rangle, S', r\right)} \\
&\Bigg[\Bigg(\underbrace{r + \gamma \max_{\langle a_1', \ldots, a_I'\rangle} Q\left(\boldsymbol{S}', \langle a_1', \ldots, a_I'\rangle | \boldsymbol{\theta}^-, \langle \boldsymbol{W}_1^-, \ldots, \boldsymbol{W}_I^-\rangle\right)}_{A \text{ constant value with the target } Q-\text{network}} \\
&\quad - Q\left(\boldsymbol{S}, \langle a_1, \ldots, a_I\rangle | \boldsymbol{\theta}, \langle \boldsymbol{W}_1, \ldots, \boldsymbol{W}_I\rangle\right)\Bigg)^2\Bigg] \\
&\approx \frac{1}{B} \sum_{b=1}^{B} \Bigg[\Bigg(r_b + \gamma \max_{\langle a_1', \ldots, a_I'\rangle} \sum_{i=1}^{I} \mathcal{N}_{\boldsymbol{\theta}^-}^{T}\left(\boldsymbol{S}_b'\right) \boldsymbol{W}_i^- \boldsymbol{\delta}\left(a_i'\right) \\
&\quad - \sum_{i=1}^{I} \mathcal{N}_{\boldsymbol{\theta}}^{T}\left(\boldsymbol{S}_b\right) \boldsymbol{W}_i \boldsymbol{\delta}\left(a_{ib}\right)\Bigg)^2\Bigg] \qquad (7)
\end{aligned}
$$

5. Asynchronous algorithm to train DGQN

The architecture of a DGQN takes the form of a single-agent RL method but can accommodate a large state and action spaces without the need of an exorbitant amount of computer memory. However, this does not mean that the proposed DGQN can be trained using only limited examples of state–action pairs. Basically, for convergence a DGQN must go through as many transition experiences as possible while training, which requires a considerable amount of time. To reduce the computation time in the present study, multiple actor-learners were mobilized in parallel, and a RL environment was copied to them so that a separate traffic simulation could be carried out for each actor-learner. At a given simulation time, each actor-learner had a traffic condition that differed from that of their counterparts, which also made it possible to avoid autocorrelations between consecutive traffic states when a single traffic simulation would be used.

A common Q-network was set up and asynchronously updated by multiple actor-learners (see Fig. 4), each of which learned from its own environment. A target Q-network was also shared by multiple actor-learners. Each actor-learner managed its own replay set unlike the original A3C algorithm wherein no replay set is used. This scheme required a greater amount of computer memory but was advantageous in facilitating the convergence. An actor-learner in an asynchronous setting should not be confused with an agent in a multi-agent RL algorithm. Basically, actor-learners must be independent of each other in that an agent's

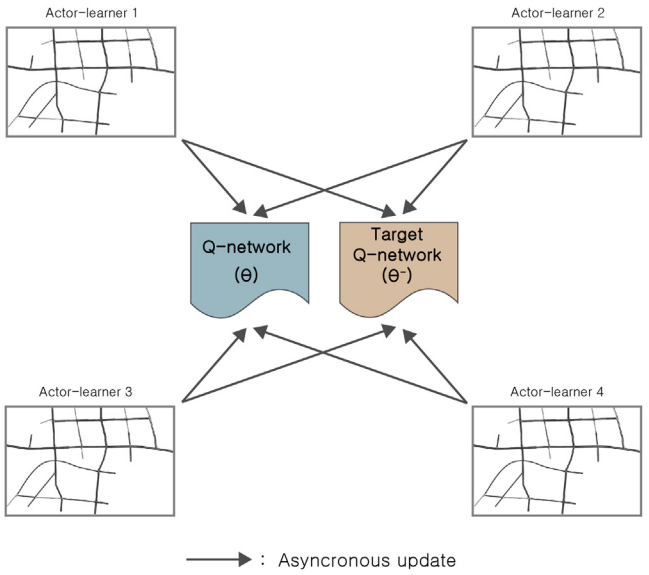


Fig. 4. The concept of asynchronous updating of a shared Q-network.

action only affects its own environment, whereas agents in a multi-agent RL algorithm must be dependent in that an agent's action changes a single shared environment. An outline of the proposed asynchronous algorithm for each actor-learner thread is shown in Fig. 5. Applying this asynchronous training scheme to the proposed DGQN that can accommodate a large number of joint actions made it possible to solve a large-scale traffic signal control problem.

The simulation experiment was performed under the following computing conditions. The main memory was 256 GB. Python main-logic and traffic-simulation software (Vissim v10.0) was implemented on two CPUs with the following specifications: AMD Ryzen Threadripper 3960X Processor @ 3.8 GHz. There were 24 available CPU cores, which fell under the maximum number of cores (=32) that the traffic simulator allows. The proposed DGQN was trained on a single GPU, which was a NVIDIA GeForce GTX 1080Ti with 11 GB of GDDR5 memory. A graphic accelerator was used to train the DGQN, whereas the animation was run on CPU cores. It took an average of 4 min to run a single episode. The total computation time for 2000 episodes was tantamount to 40 h.

6. Simulation experiments

6.1. A description of the testbed

A handicap of RL-based traffic control is that the algorithm cannot be trained in the real world. A robust simulator is necessary to train the algorithm. Vissim v10.0, a commercial traffic simulator, was chosen to mimic real-world traffic conditions. On the other hand, the testbed was chosen among real transportation networks. The testbed lies in the southwest area of Seoul, Korea as shown in Fig. 6. The network includes 15 intersections to be jointly controlled in the present study. Intersections that are not numbered indicate an unsignalized intersection or a grade-separated intersection without traffic lights. Fig. 6 also lists the current signal phases for each intersection in the morning peak hours. The reason some 4-leg intersections (①, ②, ⑨, ⑬, and ⑭) have only two or three phases is because they have a major road that either over- or under-passes a minor road. The cycle length ranges from 120 to 190 s.

The current phases were used as feasible phases for the present RL algorithm. That is, the currently available phases were implemented without a constant order for every time interval of the RL algorithm. The proposed RL-based traffic signal controller does not depend on the cycle length. The order of the signal phases varied according to traffic conditions. Thus, the controller must provide drivers with a relevant warning, so that they will be cognizant of variations in the signal order.

Table 1
Average traffic volumes in the entry links to the testbed during the morning peak hours.

Intersection	Direction	Entry volume (veh/h)
1	East-bound	81
1	South-bound	2506
2	South-bound	259
3	South-bound	169
4	South-bound	494
5	East-bound	2511
5	South-bound	352
7	North-bound	13
9	North-bound	236
10	West-bound	3107
12	West-bound	972
13	North-bound	485
14	North-bound	871
15	West-bound	231
15	North-bound	457
Fixed operation	South-bound	183
Roundabout	North-bound	575

6.2. Setting up traffic simulations

Traffic simulation requires the traffic volumes in the entry links of the testbed and the turning rates of traffic flows at the stop lines of intersections. The present experiment used input data that are compatible with real-world traffic conditions. The data were excerpted from the final report of a traffic impact analysis study that had been implemented in the testbed. The average entry traffic volumes and the average turning rates are shown in Tables 1 and 2, respectively.

The traffic volume of only the approaches entering the testbed was set as the input. Traffic volumes on the inner road segments were determined by the turning rates of the traffic flows at each stop line of intersection approaches. Each actor-learner shared a fixed set of average traffic flows at entry links and turning rates at each stop line of intersection approaches. However, the traffic volumes were varied within ±30% at the beginning of each episode to reflect the traffic conditions for different times of the day. In addition, the average turning rates of traffic volumes were periodically altered within ±30% during an episode in an effort to reflect the random fluctuations of real traffic volumes. Fig. 7 shows how traffic volumes and turning rates change for an episode. Because some entry approaches may be saturated due to randomly varied traffic volumes while implementing a RL algorithm, an episode for each actor-learner was terminated when at least one of the entry approaches was full of vehicles and could not receive a new vehicle from outside the transport network.

6.3. Setting up the hyper-parameters of an RL algorithm

Table 3 shows the hyper-parameters used to implement the present RL algorithm. The simulation time for each episode was set at 4000 s, which included 400 s of idling periods for warm-up. The traffic simulation was implemented each second of the simulation clock, and a RL update was implemented every 20 s of the simulation clock. More concretely, a RL update was carried out after 17 s from the onset of every update period, and 3 s of time

//Assume global shared parameters $(\boldsymbol{\theta}^{-}, <\boldsymbol{W}_1^{-}, \ldots, \boldsymbol{W}_I^{-}>)$ and $(\boldsymbol{\theta}, <\boldsymbol{W}_1, \ldots, \boldsymbol{W}_I>)$

Initialize replay set
Set counter $C = 0$
Implement the warm-up simulation for $T_{Initial}$

repeat

Set simulation time $T = T_{Initial}$.
Initialize traffic simulator
Choose initial state $\boldsymbol{S}$
while $T < T_{max}$ and traffic simulation is not over[1]

$\epsilon = (\varepsilon_{max} - \varepsilon_{min})\mathrm{e}^{-(C/E_i)^2} + \varepsilon_{min}$

Choose a joint action $< a_1, \ldots a_I >$ with ϵ-greedy policy using

If a uniform random number $> \epsilon$ then

[2]$< a_1, \ldots a_I > = \underset{<a_1,\ldots a_I>}{\text{argmax}} \; Q(\boldsymbol{S}, < a_1, \ldots a_I > | \boldsymbol{\theta}, <\boldsymbol{W}_1, \ldots, \boldsymbol{W}_I>) = \underset{<a_1,\ldots a_I>}{\text{argmax}} \sum_{i=1}^{I} \boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}^{T}(\boldsymbol{S})\boldsymbol{W}_i\boldsymbol{\delta}(a_i).$

Else $< a_1, \ldots a_I >$ is randomly chosen.

Proceed traffic simulation with the chosen joint phases during $(\Delta t - T_a)$

Receive new state $\boldsymbol{S}'$ and reward r

Store $\left(\boldsymbol{S}, < a_1, \ldots a_I >, \boldsymbol{S}', r\right)$ in replay set

If the size of the replay set reaches $\boldsymbol{D}$ **then** remove the samples in a FIFO fashion.

Choose B samples $\left(\boldsymbol{S}_b, < a_{1b}, \ldots a_{Ib} >, \boldsymbol{S}_b', r_b\right)$ from the replay set

[2]$y_b = r_b + \gamma \underset{<a_1',\ldots a_I'>}{\max} \sum_{i=1}^{I} \boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}^-}^{T}\left(\boldsymbol{S}_b'\right)\boldsymbol{W}_i^{-}\boldsymbol{\delta}\left(a_{ib}'\right) \; for \; \mathrm{b} = 1, \ldots, B$

Compute gradients: $d\boldsymbol{\theta} = \frac{\partial \sum_{b=1}^{B}\left(y_b - \sum_{i=1}^{I} \boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}^{T}(\boldsymbol{S}_b)\boldsymbol{W}_i\boldsymbol{\delta}(a_{ib})\right)^2}{\partial \boldsymbol{\theta}}$

$$d\boldsymbol{W}_{\mathrm{i}} = \frac{\partial \sum_{b=1}^{B}\left(y_b - \sum_{i=1}^{I} \boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}^{T}(\boldsymbol{S}_b)\boldsymbol{W}_i\boldsymbol{\delta}(a_{ib})\right)^2}{\partial \boldsymbol{W}_{\mathrm{i}}} \; for \; i = 1, \ldots, I$$

Perform asynchronous update $\boldsymbol{\theta}$, $<\boldsymbol{W}_1, \ldots, \boldsymbol{W}_I>$ using $d\boldsymbol{\theta}$ and $< d\boldsymbol{W}_1, \ldots, d\boldsymbol{W}_I >$

$\boldsymbol{S} = \boldsymbol{S}'$

$T = T + (\Delta t - T_a)$
$C = \mathrm{C} + 1$
Implement amber phase during T_a for lane groups with a signal change.
Implement the current phase during T_a for lane groups with a signal that does not change.
$T = T + T_a$

If $C \mod I_{target} == 0$ **then**

Update the target Q-network $\boldsymbol{\theta}^{-} = \boldsymbol{\theta}$, $<\boldsymbol{W}_{1\mathrm{e}}^{-}, \ldots, \boldsymbol{W}_I^{-}> = <\boldsymbol{W}_1, \ldots, \boldsymbol{W}_I>$

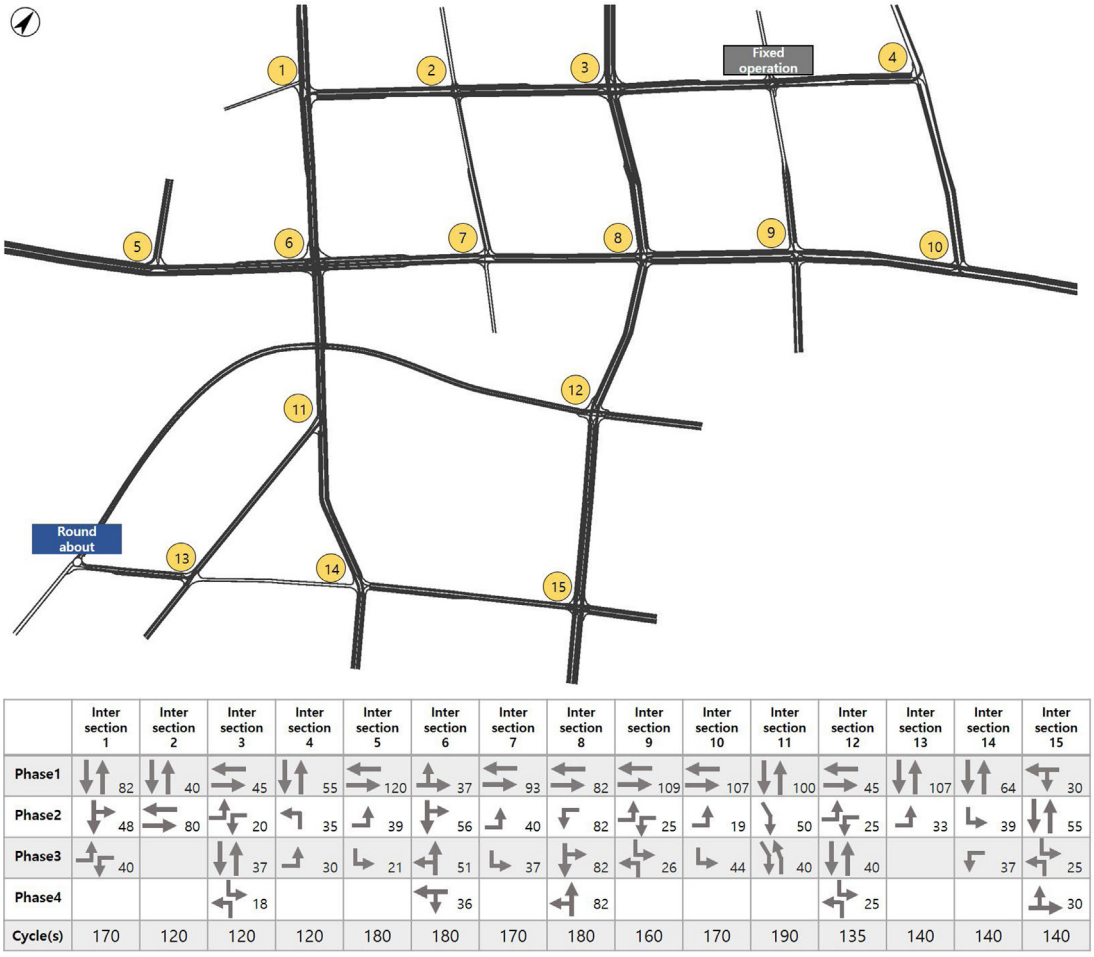
until $\epsilon \approx \varepsilon_{min}$

1. Traffic simulation is terminated when either an entry approach is full of vehicles or the total delay in a transport network during a single period exceeds the threshold (=16,000 seconds)

2. Finding the maximum is trivial based on the following relationship
$\underset{<a_1,\ldots a_I>}{\max} \sum_{i=1}^{I} \boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}^{T}(\boldsymbol{S})\boldsymbol{W}_i\boldsymbol{\delta}(a_i) = \sum_{i=1}^{I} \underset{a_i \in \{1,\ldots,\Phi\}}{\max} \boldsymbol{\mathcal{N}}_{\boldsymbol{\theta}}^{T}(\boldsymbol{S})\boldsymbol{W}_i\boldsymbol{\delta}(a_i).$

Fig. 5. Asynchronous algorithm for each actor-learner thread, which was used to train the proposed Q-network.

was provided for amber signals. The lane groups received a 3-s amber phase on the simulation clock when their current signal phase switched to another, whereas the lane groups without the phase change continued their current phase.



	Inter section 1	Inter section 2	Inter section 3	Inter section 4	Inter section 5	Inter section 6	Inter section 7	Inter section 8	Inter section 9	Inter section 10	Inter section 11	Inter section 12	Inter section 13	Inter section 14	Inter section 15
Phase1	82	40	45	55	120	37	93	82	109	107	100	45	107	64	30
Phase2	48	80	20	35	39	56	40	82	25	19	50	25	33	39	55
Phase3	40		37	30	21	51	37	82	26	44	40	40		37	25
Phase4			18			36		82				25			30
Cycle(s)	170	120	120	120	180	180	170	180	160	170	190	135	140	140	140

Fig. 6. Testbed for simulation experiments.

Table 2
Average turning rates of traffic flows during the morning peak hours.

Inter-section	East-bound			West-bound			North-bound			South-bound		
	Left turn (%)	Through (%)	Right turn (%)	Left turn (%)	Through (%)	Right turn (%)	Left turn (%)	Through (%)	Right turn (%)	Left turn (%)	Through (%)	Right turn (%)
1	83	–	17	24	–	76	–	85	15	14	58	28
2	4	80	16	9	86	5	26	29	45	59	23	18
3	11	62	27	20	73	7	65	10	25	33	35	32
4	57	53	–	–	–	–	46	55	–	44	–	56
5	13	87	–	–	82	18	–	–	–	49	–	51
6	20	6	74	55	19	26	43	46	11	11	83	6
7	11	81	8	93	–	7	–	–	100	35	14	51
8	–	81	19	3	69	28	37	23	40	19	71	10
9	6	87	7	7	87	6	68	–	32	7	–	93
10	7	93	–	–	83	17	–	–	–	67	–	33
11	–	94	6	–	–	–	–	100	–	–	53	47
12	37	45	18	26	32	42	10	59	31	25	38	37
13	71	–	29	–	–	100	16	70	14	–	59	41
14	–	–	100	46	–	54	–	82	18	15	82	3
15	–	31	69	–	74	26	11	73	16	2	81	17

Setting up the rates of exploration and exploitation is very important for the convergence of a RL algorithm. The exploration rate dwindled from the maximum (=1.0) down to the minimum (=0) while implementing a RL algorithm. The present study differentiated the exploration rate for each actor-learner. A different decaying parameter was applied to each actor-learner, as shown in Table 3. Empirical evidence showed that this scheme expedited the convergence. The target Q-network was updated for every 2500 updates of the incumbent Q-network.

7. Results from simulation experiments

The objectives of the present study were three-fold. The first objective was to confirm whether the proposed DGQN, with modifications to accommodate a large action space, could effectively manage joint traffic signal control on an area-wide scale. The objective was accomplished since a RL model with the remedy to accommodate a large action space succeeded in jointly controlling 15 intersections. The second objective was to prove the utility

Table 3
Hyper-parameters for implementing an RL algorithm.

Hyper-parameter	Description	Applied value
Δ	Time step for traffic simulation	1 s
T_{max}	Simulation period for each episode	4000 s
$T_{initial}$	Warming-up period for each episode	400 s
Δt	Time interval for RL algorithm	20 s
T_a	Amber time	3 s
ε_{max}	Initial probability of exploration	1.0
ε_{min}	Final probability of exploration	0
E_1	Decaying parameter of exploration probability for actor-learner1	2.3E6
E_2	Decaying parameter of exploration probability for actor-learner2	2.6E6
E_3	Decaying parameter of exploration probability for actor-learner3	2.9E6
E_4	Decaying parameter of exploration probability for actor-learner4	3.2E6
D	Size of replay memory set	30,000
$D_{initial}$	Initial replay memory	3000
B	Size of mini-batch	32
I_{target}	Cycle for updating target Q-function	2500 iterations

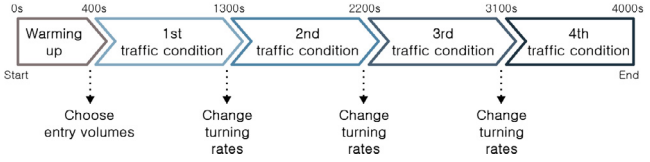


Fig. 7. Varying traffic conditions for an episode. Traffic simulation during the warm-up period was based on the average entry traffic volumes and on the average turning rates.

of adopting graph convolutions with variable adjacency matrices. To verify this objective, several reference models were set up for comparison (see Section 7.1). The last objective involved demonstrating the efficiency of asynchronous updates for a DGQN. The evidence for the utility of an asynchronous update was clarified, because we failed in leading to a convergence of the DGQN with a single environment. In theory, it would be possible for a single actor-learner to obtain a convergence with a single environment if a long computing time was allowed. However, a DGQN did not lead to a convergence without asynchronous updating, even though it was trained on the same number of episodes as the sum of each actor-learner's episodes.

7.1. Selecting reference models

The current fixed operation of traffic signals in the testbed was chosen as a baseline (see Fig. 6). A simple RL algorithm that involved a DQN composed only of FC layers was chosen as the second reference (DQN-FC). Another DQN with the original graph convolutions that used a constant adjacency matrix was established as the last reference model (DQN-OGCN). The performances of these three references were compared with that of the proposed DGQN. For fair comparisons, the proposed scheme to accommodate a large action dimension was commonly applied to the architectures of both the DQN-FC and the DQN-OGCN. Moreover, each reference network was designed so that the number of weight parameters could match that of the proposed DGQN. In particular, the former architecture of the DQN-OGCN to abstract the traffic state was the same as that of the proposed DGQN, with the exception of a parameterized adjacency matrix. For reference models and a DGQN, 600 episodes were implemented to test models trained for about 2000 episodes. For each testing episode, the same rule used in training times was also adopted to generate traffic conditions.

Besides the value-based models, an actor–critic RL algorithm based on the policy gradient was chosen as another reference for comparison. As mentioned earlier, the signal phase duration is optimized with the phase sequence fixed in an actor–critic model. This policy gradient model uses two neural networks, each of which is used to approximate a policy function and an action value function (=Q-function), respectively. The actor network corresponds to the policy function, while the critic network to the Q-function. The actor network is updated with its gradient with respect to its weights. According to the policy gradient theorem [43], when computing the gradient, the critic network that has previously been updated is used to regulate it. After updating the actor network, the critic network is updated according to the conventional Q-learning theory. These two steps alternately repeat until convergence. The critic network was set up to include the former graph convolution layers of the DGQN, and the actor network was separated for each intersection following the rule of "centralized critic and decentralized policies" [44].

In the same context of the policy gradient RL method, two fuzzy algorithms were employed for comparison. First, a naïve fuzzy system was set up for the present traffic light control problem. The delay and queue length of each lane group was chosen as input, and how much green time can be extended was chosen as the output of the system. The decision was made every time the current phase terminated. Fuzzy sets for the two input variables and the output variable were designed based on our intuitions and experiences, and fuzzy rules that designated the relationship between the input and output variables were also rationally assumed. 6 fuzzy sets were assumed for each input variable, and 6 fuzzy sets were used to represent the output variable. Correspondingly, 36 plausible fuzzy rules were made based on our insight. A neuro fuzzy system had the same fuzzy sets and rules and incorporated three functions of fuzzification, rule-based inference, and defuzzification into a neural network. The fuzzy neural network was trained on data extracted from a traffic simulator, Vissim v10.0. The two fuzzy systems sought to determine the duration of traffic signal phases with the order of phases fixed.

7.2. The convergence and test results of value-based RL algorithms

An asynchronous update was applied to training all the value-based RL models as well as a DGQN. Fig. 8 shows the convergence of the RL models. During training, an average reward was computed for every episode and plotted as implementing successive episodes. For brevity, the convergence of the first actor-learner was depicted, since those of the remaining actor-learners were not much different. The black solid line represents the moving average of the previous 10 average rewards, which clarifies the trend of convergence.

The average reward of the former two models (DGQN and DGQN-OGCN) was based on graph convolutions and converged at 0.3, whereas the average reward for the DQN-FC with FC layers

Table 4
A comparison of model performance based on the total traffic delay.

Total delay (h)	DGQN		DQN-OGCN		DQN-FC		Baseline (Current fixed operation)	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
	381.9	109.8	496.8	178.1	630.3	196.6	685.2	110.3

Table 5
A comparison of model performance based on the maximum queue length.

Maximum queue length (m)	DGQN		DQN-OGCN		DQN-FC		Baseline (Current fixed operation)	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
	83.4	13.1	101.8	24.3	109.6	17.7	113.5	8.2

led to a higher value (=0.5). This was because the traffic delays of the latter model fluctuated more than those of the two former models during training, and the reward was computed based on the change in the total delay in the testbed. Fig. 9 expounds upon this phenomenon. The total delay fluctuated more within each episode for a DQN-FC. If the total delay increased at any iteration of an episode, a DQN-FC successively received a positive reward that allowed it to recover the previous damage. On the other hand, there was no drastic change in the total delay while training a DGQN. Thus, a DGQN showed a more consistent pattern of alternating positive and negative rewards, which decreased the average rewards during training. As a result, the variance in traffic delays was smaller for a DGQN than that for a DQN-FC.

According to guidelines in the highway capacity manual, the total traffic delay and the maximum queue length were chosen to compare the performance of the proposed DGQN with those of reference models. For each model, 600 delay indices and 600 queue length indices were derived from testing episodes. Figs. 10 and 11 depict the probabilistic distributions drawn from the delay and queue length indices, respectively. Tables 4 and 5 list the numerical parameters derived from the distributions. When comparing models based on the mean of performance indices, the DGQN outperformed all other references. The DQN-OGCN ranked as the second-best model, which implies that graph convolutions were advantageous in recognizing spatio-temporal dependencies among traffic states in a transportation network. Regarding comparisons based on the variance of performance indices, the DGQN also proved to be more robust than the two other RL-based reference models, which means it resulted in the most consistent operation of traffic lights.

There was a distinct outcome with regard to the variance of the performance indices. A fixed operation had smaller variances than the two other RL-based reference models, even though it had larger average delays and queue lengths. This result is consistent with a finding by Casas [19] that the output of Q-learning when applied to traffic signal control showed a larger variance. On the other hand, a DGQN had a smaller variance in traffic delays compared with an affixed operation. In this regard, the success of a DGQN in reducing the variance in delays translates to a great enhancement. Also, a DGQN had a similar variance in maximum queue lengths compared with a fixed operation.

It should be noted that there were outliers for a DGQN in both performance indices, even though it recorded the best performance on average. This problem is common to all available RL algorithms, since the RL environment of traffic light control is intrinsically nonstationary. The traffic condition within an RL environment is affected by varying the entry traffic volumes and their dynamic route choice patterns, which are totally exogenous and cannot be known in advance. This means that the transition probabilities for a MDP are inconsistent. To overcome this complication, Padakandla and Bhatnagar [45] proposed a robust method to dynamically differentiate the state regimes of a RL environment. We are attempting to apply this method to traffic signal control on a large scale. In further studies we expect to reduce the variance in traffic delays and queue lengths.

7.3. The test results of policy-based rl algorithm and fuzzy systems

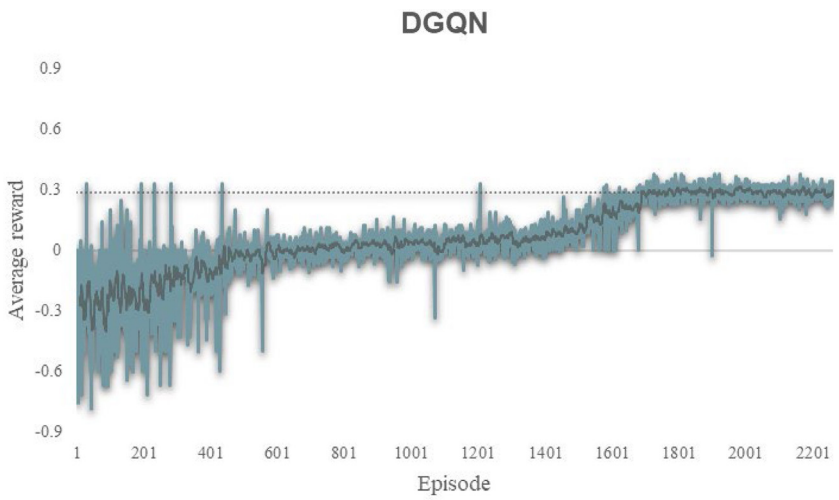
An actor–critic algorithm was trained and tested for the same environment of the value-based RL algorithms. The policy-based model did not outperform the DGQN when comparing the average delay and queue length (see Figs. 12 and 13). Tables 6 and 7 confirm the results based on numerical parameters. This might be due to the limited degree of freedom that prevented an actor–critic algorithm from changing the sequence of traffic signal phases. Nonetheless, an actor–critic algorithm succeeded in reducing the variance in the traffic delay and queue length. It is widely accepted that a policy-based RL algorithm is better than a Q-learning algorithm in reducing the variance in the reward. Our further study will focus on adjusting a policy-based RL algorithm to a large-scale discrete traffic signal control.

Regarding the test results of two fuzzy systems, Figs. 12 and 13 show that no algorithm outperformed the proposed DGQN, and the results are supported numerically in Tables 6 and 7. This outcome is compatible with our prior expectation that a rule-based AI cannot adjust to conditions out of the predefined rule set, whereas a learning-based AI responds well to them. Another reason for this outcome might stem from the fact that fuzzy reference models could not accommodate the action coordination, since they were implemented independently for each intersection. When the two fuzzy models were compared, a fuzzy neural network marginally outperformed a naïve fuzzy system, because the former was trained on simulation data but the latter was set up based solely on human intuitions and experiences. For the comparison between an actor–critic and a fuzzy neural network, the former outperformed the latter in minimizing traffic delays, but vice versa in minimizing queue lengths. Another plausible finding was that a fuzzy neural network had a variance in the queue length that was smaller than that of the DGQN.

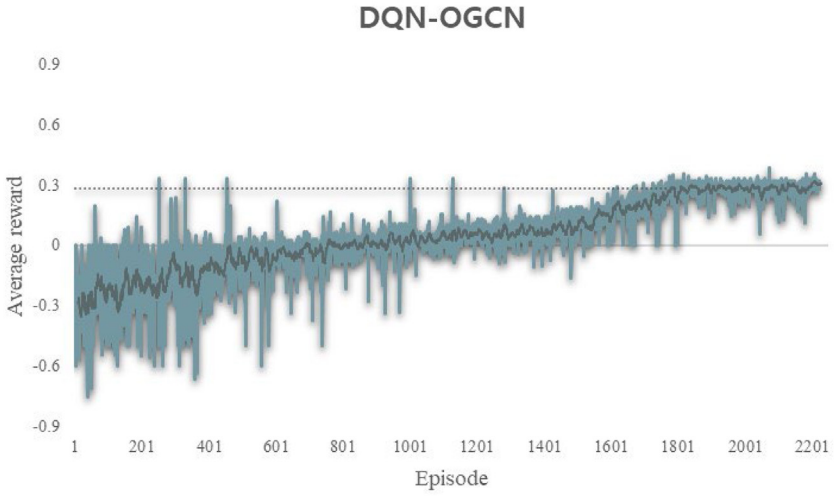
8. Conclusions and further studies

The contribution of the present study is three-fold. First, the conventional DQN architecture was revised to accommodate a large action space, which made it possible to jointly control real-world traffic lights in a fully coordinated fashion. Second, the present study adopted novel graph convolution layers parameterized with variable adjacency matrices to replace dense or 2-D convolutional layers in the conventional DQN model. The graph convolutions mimic actual traffic propagation in a large-scale transportation network in mapping the traffic state into action-values. Last, a practical way to secure the convergence of a large-scale RL model was devised by employing multiple actor learners, each of which updated the DGQN in an asynchronous manner during training. Simulation experiments for a real-world transportation network confirmed that combining these three schemes could secure the convergence of the proposed DGQN-based traffic control.

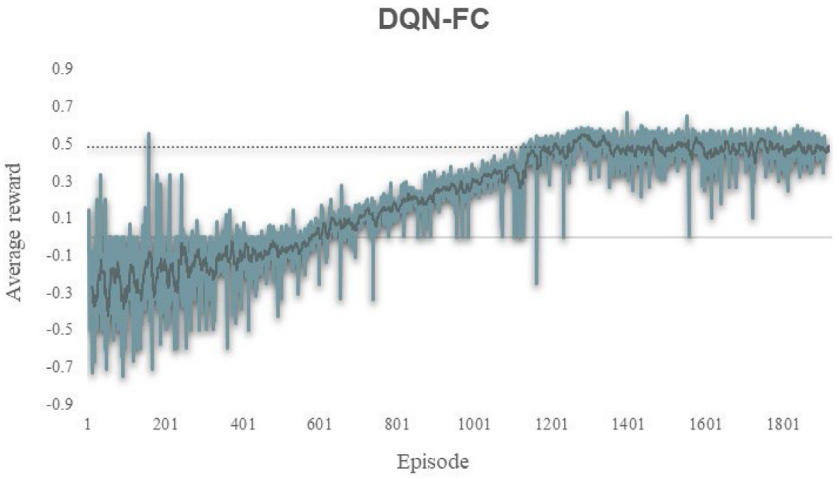
Nonetheless, the proposed approach was still hampered by at least one limitation. The most critical problem was that even



(a) Convergence in a DGQN



(b) Convergence in a DQN-OGCN



(c) Convergence in a DQN-FC

Fig. 8. Convergence in RL models.

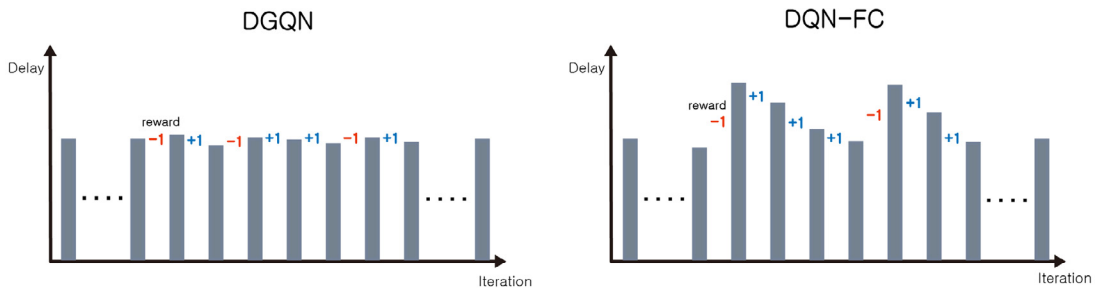


Fig. 9. The trend of rewards for different RL models.

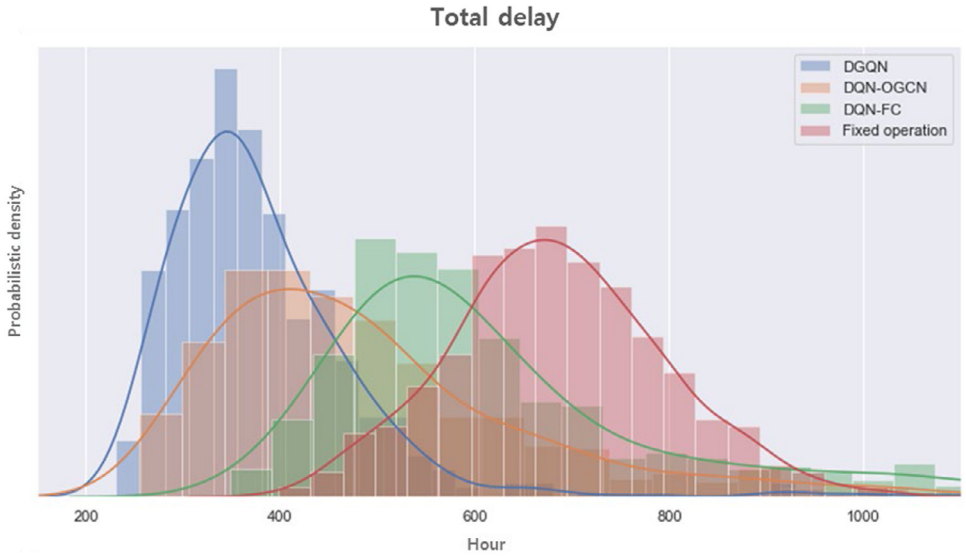


Fig. 10. Distributions of total traffic delay from different DQN models.

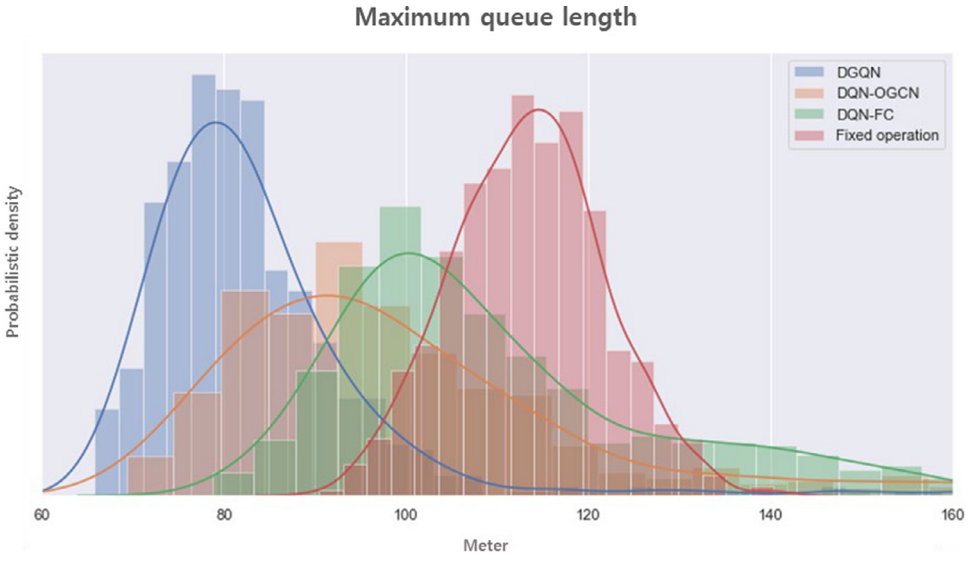


Fig. 11. Distributions of maximum queue length from different DQN models.

Table 6
A comparison of the model performance with soft actor–critic and fuzzy traffic signal controllers based on the total traffic delay.

Total delay (h)	DGQN		Soft actor–critic		Fuzzy neural net		Fuzzy system		Baseline (Current fixed operation)	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
	381.9	109.8	527.7	82.8	563.5	152.5	617.3	156.2	685.2	110.3

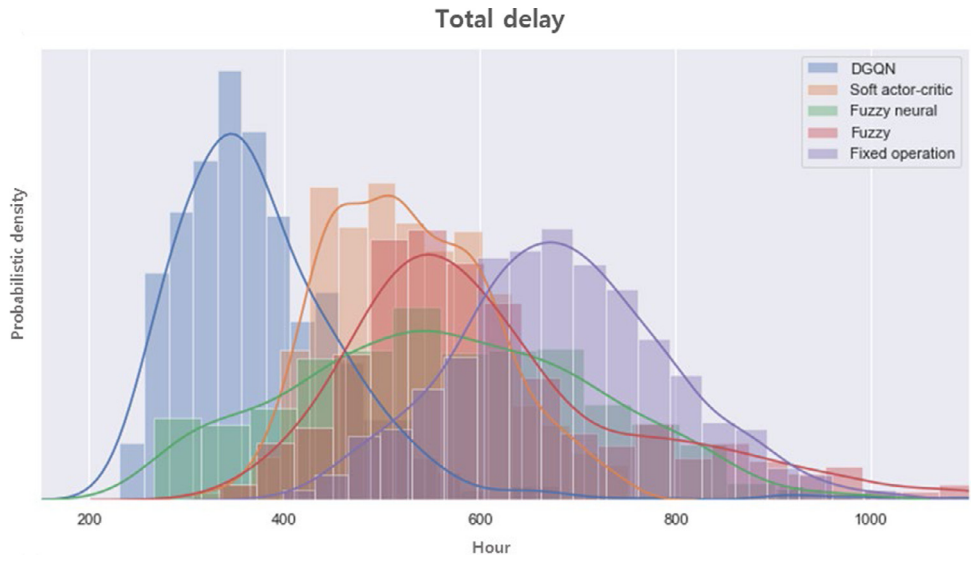


Fig. 12. Distributions of the total traffic delays from soft actor–critic and fuzzy traffic signal controllers.

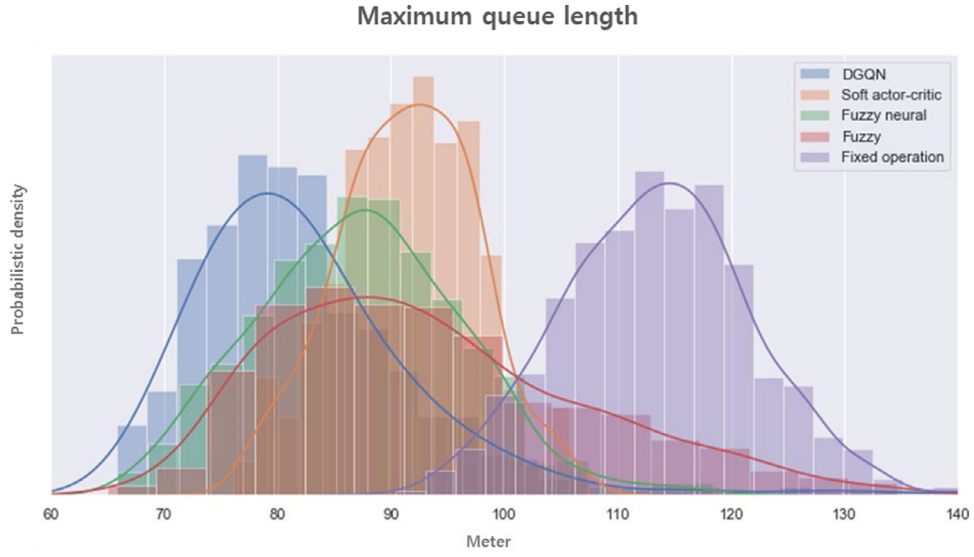


Fig. 13. Distributions of the maximum queue lengths from soft actor–critic and fuzzy traffic signal controllers.

Table 7
A comparison of the model performances of soft actor–critic and fuzzy traffic signal controllers based on maximum queue lengths.

Maximum queue length (m)	DGQN		Soft actor–critic		Fuzzy neural net		Fuzzy system		Baseline (Current fixed operation)	
	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation	Mean	Standard deviation
	83.4	13.1	91.5	6.2	87.2	9.1	93.4	13.7	113.5	8.2

though an asynchronous learning method was adopted to reduce the training time, the DGQN could not avoid failing to converge for city-wide traffic control that included hundreds or thousands of intersections. This is the main reason that multi-agent RL models prevail in academia at the expense of losing the opportunity to find a global optimum. As a practical consideration, the highway capacity manual (HCM) exemplifies that the adequate number of intersections to be jointly controlled ranges approximately from 12 and 20 for real-world applications. As a single-agent RL model, the proposed DGQN could be the best option to secure a global optimum solution to real-world traffic signal control problems. We did not argue that the proposed DGQN could resolve the curse of dimensionality when a large number of intersections must be jointly controlled. What we provide is an efficient neural network architecture that is able to accommodate any size of action-state space without an exorbitant use of computer memory. It should be noted that there is no perfect RL model to overcome the curse of dimensionality for an action-state space. The meaningful advantage of the present study is that it pioneered a practical way that depend on a single agent to find a global solution that reflects perfect coordination among jointly controlled traffic lights.

We have an important task to fulfill. As far as we could ascertain, there is no RL-based traffic signal controller that is trained and implemented in the field. It is very difficult even to adapt a RL-based controller to a real-world situation after the model has been fully trained on simulated environments. While fine-tuning

a RL model in the field, it became apparent that drivers must encounter the occasional occurrence of unexpected traffic delays due to traffic signals that a RL controller randomly determines. We are devising a methodology to fine-tune a pre-trained RL model in a supervised manner using past signal operation data. If successful, a DGQN is expected to be operational in real-world settings.

CRediT authorship contribution statement

Gyeongjun Kim: Software, Data curation, Investigation, Visualization. **Keemin Sohn:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Chung-Ang University Research Scholarship Grants in 2022, in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (2021R1A2C2003842), and in part by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (22TLRP-C148677-05).

References

[1] L. Buşoniu, R. Babuška, B. De Schutter, Multi-agent reinforcement learning: An overview, in: Innovations in Multi-Agent Systems and Applications-Vol. 1, Springer, Berlin, Heidelberg, 2010, pp. 183–221.
[2] J. Lee, J. Chung, K. Sohn, Reinforcement learning for joint control of traffic signals in a transportation network, IEEE Trans. Veh. Technol. 69 (2) (2019) 1375–1387.
[3] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, H. Xiong, STMARL: A Spatio-temporal multi-agent reinforcement learning approach for traffic light control, 2020, arXiv preprint arXiv:1908.10577.
[4] T. Nishi, K. Otaki, K. Hayakawa, T. Yoshimura, Traffic signal control based on reinforcement learning with graph convolutional neural nets, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 877–883.
[5] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, K. Harley, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning, 2016, pp. 1928–1937.
[6] M.A. Wiering, Multi-agent reinforcement learning for traffic light control, in: Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000), 2000, pp. 1151–1158.
[7] M. Steingrover, R. Schouten, S. Peelen, E. Nijhuis, B. Bakker, Reinforcement learning of traffic light controllers adapting to traffic congestion, in: BNAIC, 2005, pp. 216–223.
[8] L. Kuyer, S. Whiteson, B. Bakker, N. Vlassis, Multiagent reinforcement learning for urban traffic control using coordination graphs, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, Berlin, Heidelberg, 2008, pp. 656–671.
[9] D. Houli, L. Zhiheng, Z. Yi, Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network, EURASIP J. Adv. Signal Process. 2010 (1) (2010) 724035.
[10] F. Zhu, H.A. Aziz, X. Qian, S.V. Ukkusuri, A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multi-agent framework, Transp. Res. C 58 (2015) 487–501.
[11] S. El-Tantawy, B. Abdulhai, H. Abdelgawad, Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown toronto, IEEE Trans. Intell. Transp. Syst. 14 (3) (2013) 1140–1150.
[12] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, S. Bellemare, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[13] D. Xie, Z. Wang, C. Chen, D. Dong, July. IEDQN: Information exchange DQN with a centralized coordinator for traffic signal control, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.
[14] X. Liang, X. Du, G. Wang, Z. Han, A deep reinforcement learning network for traffic light cycle control, IEEE Trans. Veh. Technol. 68 (2) (2019) 1243–1253.
[15] T. Tan, F. Bao, Y. Deng, A. Jin, Q. Dai, J. Wang, Cooperative deep reinforcement learning for large-scale traffic grid signal control, IEEE Trans. Cybern. 50 (6) (2019) 2687–2700.
[16] E. Van der Pol, F.A. Oliehoek, Coordinated deep reinforcement learners for traffic light control, in: Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016), 2016.
[17] S. Shen, G. Shen, Y. Shen, D. Liu, X. Yang, X. Kong, PGA: AN efficient adaptive traffic signal timing optimization scheme using actor-critic reinforcement learning algorithm, KSII Trans. Internet Inf. Syst. (TIIS) 14 (11) (2020) 4268–4289.
[18] W. Genders, S. Razavi, Policy analysis of adaptive traffic signal control using reinforcement learning, J. Comput. Civ. Eng. 34 (1) (2020) 04019046.
[19] N. Casas, Deep deterministic policy gradient for urban traffic light control, 2017, arXiv preprint arXiv:1703.09035.
[20] T. Chu, J. Wang, L. Codecà, Z. Li, Multi-agent deep reinforcement learning for large-scale traffic signal control, IEEE Trans. Intell. Transp. Syst. 21 (3) (2019) 1086–1095.
[21] I. Clavera, V. Fu, P. Abbeel, Model-augmented actor-critic: Backpropagating through paths, 2020, arXiv preprint arXiv:2005.08068.
[22] T. Wang, S. Wu, Z. Wang, Y. Jiang, T. Ma, Z. Yang, A multi-featured actor-critic relay selection scheme for large-scale energy harvesting WSNs, IEEE Wirel. Commun. Lett. (2020).
[23] L. Yan, F. Xiaoping, Design of signal controllers for urban intersections based on fuzzy logic and weightings, in: Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems, Vol. 1, 2003, pp. 867–871, http://dx.doi.org/10.1109/ITSC.2003.1252073.
[24] B.M. Nair, J. Cai, A fuzzy logic controller for isolated signalized intersection with traffic abnormality considered, in: 2007 IEEE intelligent vehicles symposium, 2007, pp. 1229–1233.
[25] T. Royani, J. Haddadnia, M. Alipoor, Control of traffic light in isolated intersections using fuzzy neural network and genetic algorithm, Int. J. Comput. Electr. Eng. 5 (2013) 142.
[26] L. Zang, L. Jia, Y. Luo, An intelligent control method for urban traffic signal based on fuzzy neural network, in: 2006 6th World Congress on Intelligent Control and Automation, Vol. 1, 2006, pp. 3430–3434.
[27] N. Kumar, S.S. Rahman, N. Dhakad, Fuzzy inference enabled deep reinforcement learning-based traffic light control for intelligent transportation system, IEEE Trans. Intell. Transp. Syst. (2020).
[28] M. Abdoos, N. Mozayani, A.L. Bazzan, Hierarchical control of traffic signals using Q-learning with tile coding, Appl. Intell. 40 (2) (2014) 201–213.
[29] I. Arel, C. Liu, T. Urbanik, A.G. Kohls, Reinforcement learning-based multi-agent system for network traffic signal control, IET Intell. Transp. Syst. 4 (2) (2010) 128–135.
[30] B. Bakker, S. Whiteson, L. Kester, F.C. Groen, Traffic light control by multiagent reinforcement learning systems, in: Interactive Collaborative Information Systems, Springer, Berlin, Heidelberg, 2010, pp. 475–510.
[31] J. Jin, X. Ma, Adaptive group-based signal control by reinforcement learning, Transp. Res. Procedia 10 (2015) 207–216.
[32] J. Iša, J. Kooij, R. Koppejan, L. Kuijer, Reinforcement learning of traffic light controllers adapting to accidents, in: Design and Organisation of Autonomous Systems, 2006, pp. 1–14.
[33] J. Shin, S. Roh, K. Sohn, Image-based learning to measure the stopped delay in an approach of a signalized intersection, IEEE Access 7 (2019) 169888–169898.
[34] J. Chung, K. Sohn, Image-based learning to measure traffic density using a deep convolutional neural network, IEEE Trans. Intell. Transp. Syst. 19 (5) (2017) 1670–1675.
[35] P.G. Balaji, X. German, D. Srinivasan, Urban traffic signal control using reinforcement learning agents, IET Intell. Transp. Syst. 4 (3) (2010) 177–188.
[36] M. Abdoos, N. Mozayani, A.L. Bazzan, Holonic multi-agent system for traffic signals control, Eng. Appl. Artif. Intell. 26 (5–6) (2013) 1575–1587.
[37] M. Abdoos, N. Mozayani, A.L. Bazzan, Traffic light control in non-stationary environments based on multi agent Q-learning, in: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2011, pp. 1580–1585.
[38] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2017, arXiv preprint arXiv:1609.02907.
[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.

[40] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, Y. Fu, Image super-resolution using very deep residual channel attention networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 286–301.
[41] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, A.A. Alemi, Watch your step: Learning node embeddings via graph attention, in: Advances in Neural Information Processing Systems, 2018, pp. 9180–9190.
[42] B. Yu, Y. Lee, K. Sohn, Forecasting road traffic speeds by considering area-wide spatio-temporal dependencies based on a graph convolutional neural network (GCN), Transp. Res. C 114 (2020) 189–204.
[43] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[44] J.K. Gupta, M. Egorov, M. Kochenderfer, Cooperative multi-agent control using deep reinforcement learning, in: International conference on autonomous agents and multiagent systems, 2017.
[45] S. Padakandla, S. Bhatnagar, Reinforcement learning in non-stationary environments, 2019, arXiv preprint arXiv:1905.03970.