applied sciences

MDPI

Article

Double Deep Q-Network with Dynamic Bootstrapping for Real-Time Isolated Signal Control: A Traffic Engineering Perspective

Qiming Zheng [1], Hongfeng Xu [1,*], Jingyun Chen [1,*], Dong Zhang [1], Kun Zhang [1] and Guolei Tang [2]

[1] School of Transportation and Logistics, Dalian University of Technology, Dalian 116024, China
[2] School of Port, Waterway and Ocean Engineering, Dalian University of Technology, Dalian 116024, China
* Correspondence: hfxu@dlut.edu.cn (H.X.); chenjy@dlut.edu.cn (J.C.)

Abstract: Real-time isolated signal control (RISC) at an intersection is of interest in the field of traffic engineering. Energizing RISC with reinforcement learning (RL) is feasible and necessary. Previous studies paid less attention to traffic engineering considerations and under-utilized traffic expertise to construct RL tasks. This study profiles the single-ring RISC problem from the perspective of traffic engineers, and improves a prevailing RL method for solving it. By qualitative applicability analysis, we choose double deep Q-network (DDQN) as the basic method. A single agent is deployed for an intersection. Reward is defined with vehicle departures to properly encourage and punish the agent's behavior. The action is to determine the remaining green time for the current vehicle phase. State is represented in a grid-based mode. To update action values in time-varying environments, we present a temporal-difference algorithm TD(Dyn) to perform dynamic bootstrapping with the variable interval between actions selected. To accelerate training, we propose a data augmentation based on intersection symmetry. Our improved DDQN, termed D3ynQN, is subject to the signal timing constraints in engineering. The experiments at a close-to-reality intersection indicate that, by means of D3ynQN and non-delay-based reward, the agent acquires useful knowledge to significantly outperform a fully-actuated control technique in reducing average vehicle delay.

Keywords: double deep Q-network; traffic signal control; traffic simulation; reinforcement learning

Citation: Zheng, Q.; Xu, H.; Chen, J.; Zhang, D.; Zhang, K.; Tang, G. Double Deep Q-Network with Dynamic Bootstrapping for Real-Time Isolated Signal Control: A Traffic Engineering Perspective. *Appl. Sci.* **2022**, *12*, 8641. https://doi.org/10.3390/app12178641

Academic Editor: Luigi Fortuna

Received: 28 July 2022
Accepted: 25 August 2022
Published: 29 August 2022

1. Introduction

Isolated signal control refers to operating traffic signals at an intersection in isolation, to serve local road users without considering vehicle progression and operational performance at adjacent intersections. Signal operations have a distinguishable impact on intersection performance. To achieve better intersection performance, it is strongly preferred to adjust the green times for vehicle phases in real time to meet time-varying vehicle demand. For a long time, as well as real-time coordinated signal control, real-time isolated signal control (RISC) has been a research interest in the field of traffic engineering.

Like some control problems in other fields [1], RISC problems involve a kind of time-varying dynamics about the signals and vehicles at an intersection. To solve RISC problems, researchers usually build optimization models or define logic rules, based on their understanding of intersection's dynamics. In the language of statistics, however, the dynamics discovered and explained by researchers are merely the limited samples of the population. Quite often, the underlying reason for unsatisfactory intersection performance is that the optimization models or the logic rules fail to cope well with the dynamics beyond their reach.

Reinforcement learning (RL) is a machine learning paradigm for solving sequential decision problems. RL is learning by repeatedly interacting with environment in a trial-and-error manner. Concretely, an RL agent senses the state of its environment, and takes an action to affect the environment; then the environment transitions the state and yields a

reward; the agent utilizes the reward to improve its policy, i.e., the mapping from states to actions. Let $R_{t+1}, R_{t+2}, R_{t+3}, ...$ denote the sequence of rewards after time step t. The agent aims to seek an optimal policy π^* that maximizes the expected cumulative discounted rewards: $\pi^* = \arg\max_\pi \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots\right]$, where $\gamma \in [0, 1]$ is the discount rate [2].

It is feasible to apply RL to solve RISC problems. First, green time adjustment is made every a period of time. Current decision affects the situations facing subsequent decisions and the intersection performance in future. Obviously, RISC problems are sequential decision problems. Second, high-resolution vehicle data, e.g., location and speed, become more available as advanced traffic detection systems come into use. Such data can be used to construct the state representation and rewards for RL. Third, traffic simulation tools allow us to reproduce nearly all real-world intersection's dynamics on computers. It is inexpensive and not time-consuming for an RL agent to collect massive samples of intersection's dynamics during training.

Given the aforementioned shortcomings of the optimization models and logic rules, it is necessary to apply RL to solve RISC problems. First, an RL agent can learn knowledge, known or unknown to humans, about how to time signals to achieve better intersection performance. Second, the agent can extensively explore the state-action space of RISC problems and try to iteratively improve its policy. This leads to the learned knowledge evolving toward super-human level.

In this study, we concern ourselves with four things:

1. to profile a RISC problem that fully satisfies the signal timing constraints from the engineering perspective;
2. to compare the applicability of RL methods to the RISC problem, and choose a well-suited one;
3. to guide the definition of RL elements with traffic engineering expertise;
4. to improve the chosen RL method in terms of training settings to accommodate the RISC problem.

Our efforts in these things induce the proposal of an RL solution, termed Double Deep Dynamic Q-network, i.e., D3ynQN.

The remainder of this paper is organized as follows. In Section 2, we review previous studies and present our research focuses. In Section 3, we profile the single-ring RISC problem. In Section 4, we choose a well-suited RL method to solve the single-ring RISC problem. In Sections 5 and 6, we define RL elements and RL training settings. In Section 7, we conduct experiments at a virtual intersection modeled by Vissim. Finally, we present conclusion and discuss the possibility of applying the RL-based signal control in practice.

2. Literature Review

The early stage of energizing signal control with RL can be traced back to 2001 [3]. From 2001 to 2015, tabular RL methods found favor with a lot of studies. The success of deep Q-network (DQN) [4] in 2015 opened the door to deep RL methods, which fueled a new round of development of such an energization.

2.1. Agent Deployments

We focus on the research progress of isolated intersection signal control. In previous studies, it was common to deploy an RL agent for an intersection. Many studies were specific to the scenario of multiple intersections. However, in some of them, the agents perceived and controlled local intersections without information-sharing. Thus, the proposed methods in such studies were also applicable to isolated signals. These studies were included in the scope of literature review too.

It is believed that the mutual effects between multiple agents are intractable [5,6]. First, in the eyes of an agent, other agents' behavior is a part of its environment's dynamics. The policy improvement process will make this part non-stationary and difficult to learn.

Second, the learning progress of each agent can be inconsistent. For example, an agent can suffer from over-fitting, meanwhile another is still under-fitting to the dynamics.

2.2. Training Environments

Traffic simulation tools, e.g., Vissim, SUMO, CityFlow, and Aimsun, were widely used to build virtual intersections. Vehicle demand level during training was unchangeable over time [7–32], or varied by time of day [13–19,33–41]. Vehicle turning ratios were typically fixed. Pedestrian demand was rarely taken into consideration [20,41].

Most if not all studies employed single-ring phasing to serve vehicle phases. The signal timing constraints in engineering, such as fixed phase sequence, minimum greens, maximum greens, yellow change intervals, and red clearance (sometimes called all-red) intervals, were more or less overlooked.

In particular, waiving fixed phase sequence was unacceptable to traffic engineers, though it made training easy—given that an agent can switch to any phase anytime, a straightforward and good policy is to just activate the phase with the longest queue at each time step.

2.3. RL Methods

Model-free RL methods were far preferred over model-based RL methods. To represent state in more detail, model-free methods must be able to deal with large continuous state space. As a result, tabular methods fell out of favor [8,10] and approximate methods gained popularity. Specifically, value-based methods with deep neural networks (DNNs) as approximators, were of great interest to many researchers [17–39]. However, the underlying reasons behind these preferences were not well discussed.

2.4. Action Definitions

As shown in Table 1, the actions taken by an agent were as follows: (1) to choose a vehicle phase to display green with a specific duration; (2) to choose the green time for current vehicle phase; (3) to determine whether or not to end current vehicle phase; and (4) to adjust the green time for all vehicle phases in next cycle. The time intervals between two successive actions were usually not constant.

Table 1. Literature groups with regard to RL elements.

Item	Option	Literature Count	Literature Number
Action taken by an agent	To choose a vehicle phase to display green with a specific duration	18	[9,11–13,15,21,26–29,31–34,36,37,39,40]
	To choose the green time for current vehicle phase	4	[10,17,23,41]
	To determine whether or not to end current vehicle phase	8	[7,16,18–20,24,25,38]
	To adjust the green time for all vehicle phases in next cycle	5	[8,14,22,30,35]
Vehicle-specific performance measure used to construct rewards	Number of already served vehicles	14	[12,13,17,18,20–23,28,31,33,34,38,39]
	Wait time of already served vehicles	2	[35,38]
	Number of vehicles to be served	23	[7,8,11,13,15,17,18,20,21,24,25,27–29,31–34,36–39,41]
	Wait time of vehicles to be served	12	[8–10,14–17,22,26,30,38,40]
	Speed of vehicles to be served	2	[19,29]
Vehicle-related feature encoding for state representation	Point-based mode	3	[10,15,25]
	Grid-based mode	8	[7,15,25,26,28–30,38]
	Block-based mode	27	[8,9,11–14,16–24,27,31–41]

In fact, option (1) was unacceptable to traffic engineers, because it led to variable phase sequence and gave road users an uncomfortable perception of signal control. Option (4) adjusted signal timing too infrequently and could hardly meet the short-term fluctuations in traffic demand.

2.5. Reward Definitions

The vehicle-specific performance measures that were processed to define the rewards included: (1) the number of already served vehicles; (2) the wait time of already served vehicles; (3) the number of vehicles to be served; (4) the wait time of vehicles to be served; and (5) the speed of vehicles to be served. Note that these measures sometimes were used in combination. Typically, a reward equaled the change of the measurements between two successive actions. The most-used discount rate was 0.99.

To be frank, the rewards associated with vehicles to be served (i.e., option (3–5)) may not clearly encourage and punish agent's behavior, because they were greatly influenced by vehicle arrivals, which were independent of signal operations.

2.6. State Representations

As the key components of state, the vehicle-related features were encoded in a point-based mode, a grid-based mode, or a block-based mode (see Table 1). The point-based mode utilized conventional point detectors, such as inductive loop, to collect features like gap time or occupancy time. The grid-based mode divided roadway space into many grids. Each grid was several meters in length to contain at most one vehicle. The features collected in a grid included vehicle's presence, speed, and so forth. Quite often, the grid-based mode was used in combination with convolutional neural networks. The block-based mode divided roadway space into a few blocks. Each block was tens of meters in length to contain multiple vehicles. The features collected in a block included vehicles' count, lengths, speeds, wait times, and so forth. Note that literature [15,25,38] combined the grid-based and block-based modes to construct hybrid state features.

Basically, the point-based mode failed to capture rich spatial information about vehicles. The grid-based mode saved a lot of effort for feature engineering, whereas the block-based mode required useful features to be handcrafted. Compared with the block-based mode, the grid-based mode led to more informative state representation but made training more difficult.

2.7. Summary of Shortcomings

There were five primary shortcomings in the previous studies that motivated us to conduct this study:

1. A lack of comparison between RL methods in terms of the applicability to signal control problems. The previous studies often directly specified the RL methods they used, without considering the characteristics of signal control problems.
2. A lack of attention to the time-varying nature of traffic environments and its influence on reward discounting and action-value updating. The previous studies usually defined the time-step size as the variable interval between actions, and discounted rewards every time step.
3. A lack of utilization of intersection symmetry to benefit the RL training. The exception is Zheng et al. [37]. However, in his work, the symmetry property was used to design neural network in advance, instead of to expedite training.
4. The neglect of some signal timing constraints, especially the fixed phase sequence and maximum greens. For safety reasons, a signal control technique not abiding by such constraints can hardly implemented in practice.
5. The test-bed intersections were not close-to-reality in terms of vehicle turning and pedestrians. Specifically, the previous studies usually set fixed vehicle turning ratios and had no pedestrian demand. This widened the gap between virtual and real intersections.

3. Single-Ring RISC Problem

As a moderate attempt to bring the traffic engineering considerations into play, this study concentrates on the RISC problem with single-ring phasing. We start with the concept of ring. In the language of traffic engineering, a ring organizes the signal phases at an intersection to display green in a sequence. Single-ring phasing controls multiple non-conflicting vehicle movements with a vehicle phase and allows the vehicle phases to be served one at a time. For a four-leg intersection with protected left-turn movements and permitted right-turn movements, there are two common options for the single-ring phasing, as shown in Figure 1. The leading left-turn phasing is much more popular than the lagging one [42].

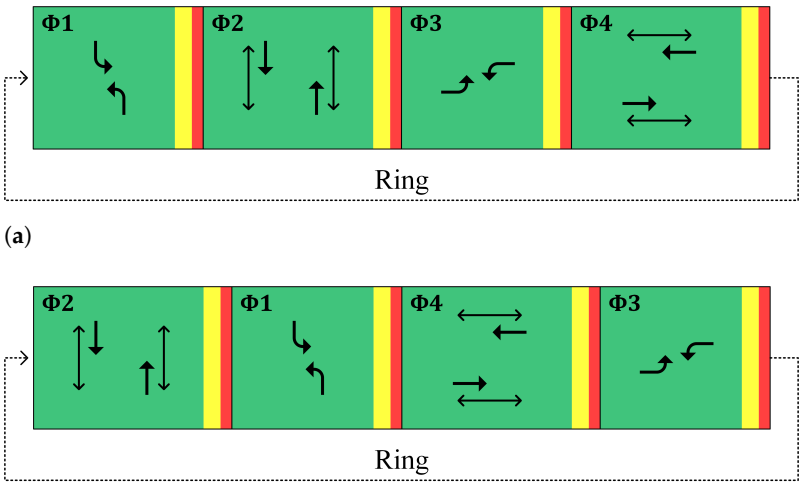

Figure 1. Common options for the single-ring phasing. Φ represents vehicle phase number, solid one-way arrows represent protected vehicle movements, and Solid two-way arrows represent pedestrian movements. Subfigure (**a**) shows the leading left-turn phasing, and (**b**) the lagging left-turn phasing.

We profile the single-ring RISC problem as follows: current vehicle phase varies its green time between the minimum green and maximum green, with the step size of one second. Green time is followed by yellow change and red clearance intervals. When the red clearance interval ends, the current vehicle phase rests in red and the next vehicle phase in the sequence starts to display green. Phase skipping is prohibited. A pedestrian phase is timed using the minimum green for its adjacent, parallel through vehicle phase.

There are many signal timing constraints that must be defined in advance. Each plays its important role in terms of safety or efficiency. Given the neglect of them in previous studies, here we introduce them as follows.

The minimum green reduces the risk of rear-end crashes and assures a certain number of queued vehicles to be served. The maximum green limits the traffic handling capacity given to a vehicle phase and prevents the road users of any other movement from experiencing excessively long red time. The yellow change interval allows time for drivers to make a decision of stopping behind or passing through stop lines before the onset of red. The red clearance interval separates the conflicts between the current vehicle phase and the next phases. The phase sequence is pre-determined at all times of the day to avoid driver confusion and reduce the risk of red-light-running violations.

4. RL Method Selection

Previous studies often specified their RL method without detailed motivations. Although it is impractical to experimentally investigate all RL methods, here we qualitatively analyze different kinds of methods and pick out a well-suited one for the single-ring RISC problem.

1. Single-agent methods versus multi-agent ones. The nature of RISC problems is to cyclically determine the green time for each vehicle phase at an intersection. As required by the single-ring phasing, different vehicle phases are prohibited from displaying green simultaneously. It means that there are no concurrent decision-making processes. Hence, a single agent deployed for the intersection is capable enough of dealing with the single-ring RISC problem.
2. Model-free methods versus model-based ones. A road user's behavior and other road users' reaction to his behavior are both stochastic. All the road users at an intersection have a highly stochastic impact on the state transitions and the resulting rewards. As a consequence, the intersection's dynamics are too complicated to be thoroughly modeled by human beings. Therefore, model-free RL methods are more sensible options for the single-ring RISC problem. In this way, the agent relies not on prior models but on massive trial-and-error experiences to predict the states and rewards.
3. Value-based methods versus policy-based ones. Regardless of the actions defined, the action space is obviously discrete. In contrast to other action options, choosing the green time for current vehicle phase often leads to a larger action space. Even so, there are only dozens of actions in it. In any case, the agent faces a discrete and small action space, which can be effectively handled by value-based RL methods.
4. Off-policy methods versus on-policy ones. We believe the policy being learned to solve the single-ring RISC problem is greedy. That means a state is mapped onto a deterministic action rather than a probability distribution over actions. However, the policy of behavior during training should be stochastic to ensure sufficient exploration of state-action space. This justifies the choice of off-policy RL methods, in which the learned policy naturally differs from the behavior policy.
5. Deep RL methods versus Tabular ones. The high-resolution vehicle data enable us to build a large continuous state space for the single-ring RISC problem, which can be handled only by approximate RL methods. DNNs are seen as universal function approximators [43]. The deep RL methods, using DNNs to approximate value function, minimize the effort of handcrafting state features and create an opportunity to apply RL in an end-to-end manner.

DQN is an RL method that can learn to play many distinct Atari video games at human-level performance [4]. Among the prevailing RL methods, only DQN and its variants simultaneously satisfy the aforementioned requirements on solving the single-ring RISC problem, i.e., being a model-free, value-based, off-policy, and deep RL method.

Most variants of DQN address the cases that exist only in particular RL tasks. However, they are usually not our case. For example, Dueling DQN is specific to the cases where actions do not significantly affect the environment in some states [44]. However, in our case, the agent decides the green time for phases. An improper action can cause obvious green starvation or overflow queue. Prioritized DQN is specific to the cases with sparse rewards [45]. However, in our case, the reward is the number of vehicle departures and is fairly dense. It is reported that these variants unexpectedly under-performed the original DQN in some tested tasks. That means, using such DQN variants may be counter-productive in the cases without the specific characteristics.

Double deep Q-network (DDQN) is another variant of DQN. It addresses the issue of over-estimating action values, by decoupling the selection and evaluation of bootstrap action. Theoretically, such over-estimations are proven to be ubiquitous when using DQN, regardless of the RL tasks involved. Experimentally, DDQN outperformed DQN in almost all Atari games, confirming its universality among diverse RL tasks. The more the actions in action space, The greater the advantage of DDQN [46]. Regarding our single-ring RISC problem, the dozens of actions allow full play to DDQN.

Taken together, DDQN is our final choice of the basic RL method to be improved.

5. RL Elements

5.1. Reward

5.1.1. Limitation of Defining Reward with Vehicle Delay

For the single-ring RISC problem, the most-used operational objective is to minimize vehicle delay at the intersection. Delay data can be collected every time a vehicle crosses the stop lines. It seems to be the best choice to define reward based directly on vehicle delay. However, in doing so, the short-term rewards may encourage inefficient green utilization due to the following reasons.

As shown in Figure 2, during the red time of a vehicle phase, the earlier a vehicle arrives, the closer it is to the stop line in the standing queue, but the more wait time it experiences, and the greater delay it reports when crossing the stop line. In the eyes of the agent, the earlier vehicles in the standing queue contribute to the worst portion of the rewards. After the standing queue discharges, the rewards become better because the newly arrived vehicles cross the stop lines without delay. In this regard, the agent is reluctant to end the current vehicle phase with no continued demand, as the short-term rewards of extending the current vehicle phase are always better than those of starting the next vehicle phase.

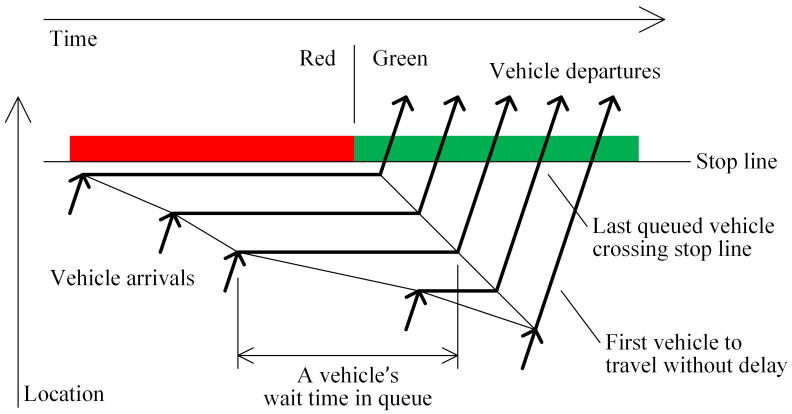

Figure 2. Basic relationship between vehicle arrivals and departures.

We admit that the agent can eventually correct this behavior preference if it is detrimental to the long-term rewards. However, such a correction will increase the time and cost of training effort.

In addition, giving the upper bound of vehicle delay is non-trivial. Hence it is hard to appropriately normalize the delay-based reward in implementation wise.

5.1.2. Proposed Reward Definition

To avoid the above limitation, we set the agent's goal as serving more vehicles within less amount of time. A vehicle is regarded as being served when its front end crosses the stop lines. Accordingly, the reward is defined as the count of vehicle departures at the intersection between two successive time steps, given by:

$$R_{i+1} = \sum_{p=1}^{P} NOV_{i,i+1}^{p} \tag{1}$$

where R_{i+1} is the reward at time step $(i + 1)$; $NOV_{i,i+1}^{p}$ is the number of vehicles departing from the stop lines, on the p^{th} approach lane of the current vehicle phase, from time step i to time step $(i + 1)$; and P is the number of approach lanes of the current vehicle phase.

Normally, there are at most one vehicle departing from an approach lane in a second. So the value of reward ranges from zero to the maximal number of approach lanes on a vehicle phase. We normalize the reward by dividing by its maximum value.

Regarding the relation of our reward to vehicle delay, an important evidence suggests that maximizing the time-weighted vehicle departures can lead to the minimization of total vehicle delay at a road network [47].

We believe the reward defined is training-friendly and can clearly encourage and punish the agent's behavior. The discounted setting of reward motivates the agent to obtain more positive rewards as early as possible [2]. In the language of traffic engineering, the agent will attempt to reduce the green time that is inefficiently used, and give more green time to vehicle phases with sufficient demand and more approach lanes. On the other hand, such reward is not affected by vehicle arrivals if queuing exists. Formally, given a state and an action taken in this state, the distribution of the resulting reward is stationary, regardless of vehicle arrivals.

5.1.3. Data Availability and Implementation

At the simulated virtual intersection, we get the reward data (i.e., the vehicle departures at stop lines) by configuring a virtual "Data Collection Point" at each stop line to count vehicle departures. Specifically, we access to all the "Data Collection Points" at the intersection, to fetch the cumulative number of vehicles passing through them since the start of current simulation run. Eventually, the reward is just the difference of the cumulative number of passing vehicles in a second.

At real intersections, the reward data can be readily collected by conventional traffic detection systems, such as the existing inductive loop detectors at stop lines.

5.2. Action

5.2.1. Limitation of Defining Action as Extending or Ending a Phase

It is common practice for traffic engineers to make a decision of extending or ending current vehicle phase on a second-by-second basis. However, we do not define action in that way, due to a thorny issue it brings to training. Specifically, in the early stage of training, the agent selects actions almost at random, to explore state-action space as much as possible. A green time that is comparatively long can be visited only if the agent decides to extend the current vehicle phase many times in a row. The possibility of doing so is extremely low. Consequently, there will be very few samples associated with long green times for the agent to learn.

5.2.2. Proposed Action Definition

To avoid the above limitation, we define the action as determining the remaining green time for current vehicle phase at the end of its minimum green, which is an integer value not greater than the difference between the minimum green and maximum green, given by:

$$A_i \in \{a \in \mathbb{Z} : 0 \leq a \leq G_{\max} - G_{\min}\} \tag{2}$$

where A_i is the action at time step i; a is an optional action; and $G_{\min}$ and $G_{\min}$ are the maximum and minimum greens for the current vehicle phase, respectively.

As a consequence, the green time for a vehicle phase is the sum of the minimum green and the action selected, which varies from cycle to cycle. Compared with the decision points earlier in a cycle, e.g., the start of green, the end of minimum green enables an action to be selected according to the latest state.

The action defined meets the requirement of fixed phase sequence, while posing a great challenge to training. In some circumstances, the agent has to learn to shorten the green time given to the current vehicle phase even with sufficient demand, so as to start the subsequent vehicle phases earlier. This is saying that the agent must learn to make short-term sacrifices for long-term benefits, and select actions in a more holistic and far-sighted manner.

5.2.3. Implementation

At the simulated virtual intersection, the selected actions are executed by directly controlling the change of signal indication of current virtual "Signal Phase" every second.

At real intersections, the selected actions can be executed by accessing to local signal controllers.

5.3. State

At real-world intersections, lane changes are permitted on the upstream links and approach tapers, but are prohibited on the approach lanes. The approach lanes are grouped to serve vehicles in compliance with the signal indications of specific vehicle phases. When constructing state representation, we intend to spatially differentiate between the approach lanes and upstream link, and between the approach lane group of one phase and that of others. By doing so, the agent has great potential to learn the intersection's dynamics in different portions of roadway space.

5.3.1. State Definition

A state observation zone that is too long may slow training because it introduces too much information irrelevant to short-term state transitions; one that is too short may result in the deficiency of essential information for state and reward predictions.

On each intersection leg, the state observation zone should cover the entire approach lanes, the entire approach taper, and a portion of the upstream link, as illustrated in Figure 3. In this study, we set the length of the state observation zone as 150 m.

We represent state in a grid-based mode. The state observation zone is classified into phase-related grids and phase-unrelated grids. Each grid is *LEN* meters long and is as wide as the lane width. In this study, *LEN* equals 4 to ensure that a grid cannot be occupied by two vehicles at the same time. As shown in Figure 3, the phase-related grids start from the stop line, and cover the portion of the approach lane length that is divisible by *LEN*. The phase-unrelated grids follow the phase-related grids and cover the rest of the state observation zone.

On the bth intersection leg, the state features of the phase-related grids on the uth approach lane group are organized as a three-dimensional state matrix $MAT^{b,u}$, given by:

$$MAT^{b,u} = \begin{pmatrix} f_{1,1}^{b,u} & f_{1,2}^{b,u} & \cdots & f_{1,C^{b,u}}^{b,u} \\ f_{2,1}^{b,u} & f_{2,2}^{b,u} & \cdots & f_{2,C^{b,u}}^{b,u} \\ \vdots & \vdots & \ddots & \vdots \\ f_{H^{b,u},1}^{b,u} & f_{H^{b,u},2}^{b,u} & \cdots & f_{H^{b,u},C^{b,u}}^{b,u} \end{pmatrix} \tag{3}$$

where $H^{b,u}$ and $C^{b,u}$ are the numbers of rows and columns in the matrix, respectively; and $f_{h,c}^{b,u}$ is the state feature of the phase-related grid at the h^{th} row and c^{th} column, which is a 3-tuple given by:

$$f_{h,c}^{b,u} = \left\langle o_{h,c}^{b,u}, v_{h,c}^{b,u}, g_{h,c}^{b,u} \right\rangle \tag{4}$$

If the front end of a vehicle is present in a phase-related grid, $o_{h,c}^{b,u}$ is set to 1 and $v_{h,c}^{b,u}$ is set to its immediate speed normalized by the posted speed limit. Otherwise, $o_{h,c}^{b,u}$ and $v_{h,c}^{b,u}$ are both set to 0. If the vehicle phase related to the grid displays green, $g_{h,c}^{b,u}$ is set to 1, otherwise 0.

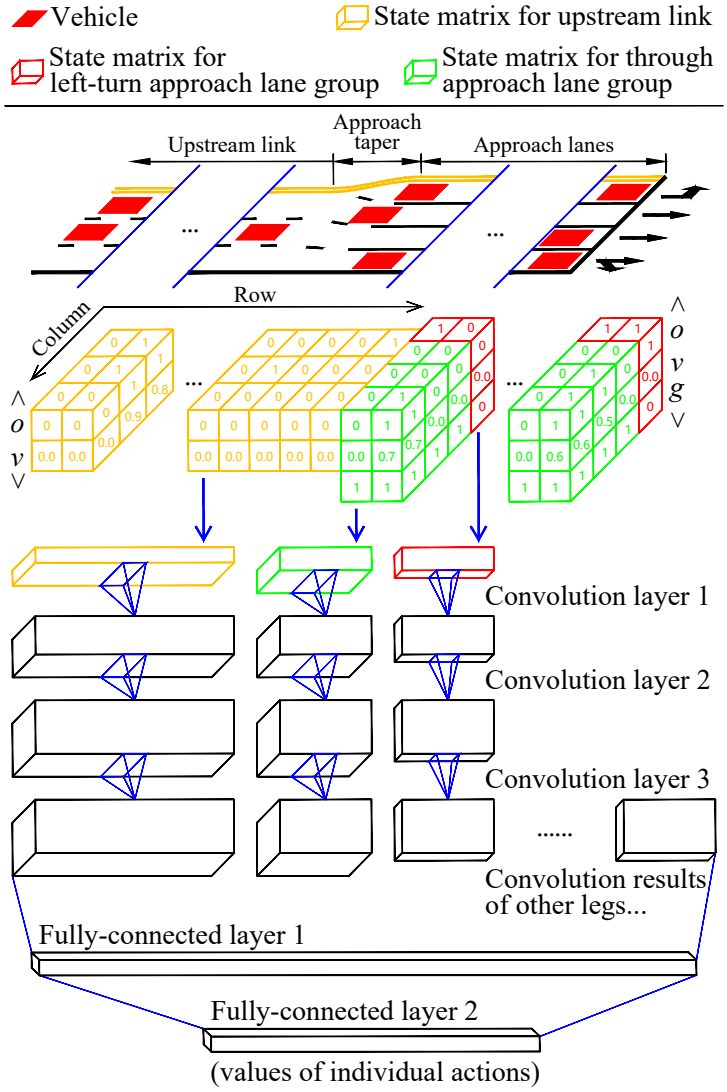

Figure 3. State representation and neural network structure.

Similarly, the state features of the phase-unrelated grids on the bth intersection leg are organized as a three-dimensional state matrix, given by:

$$MAT^b = \begin{pmatrix} f^b_{1,1} & f^b_{1,2} & \cdots & f^b_{1,C^b} \\ f^b_{2,1} & f^b_{2,2} & \cdots & f^b_{2,C^b} \\ \vdots & \vdots & \ddots & \vdots \\ f^b_{H^b,1} & f^b_{H^b,2} & \cdots & f^b_{H^b,C^b} \end{pmatrix} \tag{5}$$

where H^b and C^b are the numbers of rows and columns in the matrix, respectively; and $f^b_{h,c}$ is the state feature of the phase-unrelated grid at the hth row and cth column, which is a 2-tuple given by:

$$f^b_{h,c} = \left\langle o^b_{h,c}, v^b_{h,c} \right\rangle \tag{6}$$

$o^b_{h,c}$ and $v^b_{h,c}$ are set in the same way as $o^{b,u}_{h,c}$ and $v^{b,u}_{h,c}$ for the phase-related grids.

Lastly, our state consists of the state matrices for the phase-related and phase-unrelated grids on all the intersection legs, given by:

$$S_i = \Big(MAT^{1,1}, MAT^{1,2}, \ldots, MAT^{1,U^1}, MAT^{1}, \\ MAT^{2,1}, MAT^{2,2}, \ldots, MAT^{B,U^B}, MAT^{B} \Big) \tag{7}$$

where S_i is the state at time step i; U^b is the number of approach lane groups on the bth intersection leg; and B is the number of intersection legs.

5.3.2. Data Availability and Implementation

At the simulated virtual intersection, we construct the state representation on a vehicle-by-vehicle basis. We first fetch the coordinates and speed of each vehicle at the virtual intersection. We then locate the grid occupied by each vehicle, based on its coordinates relative to the downstream stop line. Finally, we set the state features of that grid in null state matrices filled with zeros, accroding to the descriptions of Equations (4) and (6).

At real intersections, each vehicle's coordinates and speed can be collected by emerging data-fusion systems based on radar and video detection. Although there are issues about missing and faulty data at present, we believe they would be addressed in the era of vehicle-to-infrastructure communication.

5.3.3. Neural Network Structure

The number of columns in a state matrix, equaling the number of lanes on an upstream link or in an approach lane group, is very small compared with the applications in computer vision field. To minimize the loss of information, we take a page from the convolutional network structure of DDQN, but reduce its filter sizes and strides, and add padding operation.

The exact network structure is shown schematically in Figure 3. Each state matrix is handled by 3 convolution layers. The first hidden layer convolves 32 filters with stride 1. The second and third hidden layers both convolve 64 filters with stride 1. The filter size is 3×1 if the matrix corresponds to a single lane, and is 3×3 otherwise. Zero-padding is added before each convolution operation. The convolution results of all the state matrices for the intersection are concatenated and fed to a fully-connected hidden layer with 512 units. This is followed by the output layer, which is a fully-connected linear layer with a single unit for the value of each action. Rectifier Linear Units (ReLU) [48] are inserted into all the hidden layers.

6. RL Training Settings

DDQN is based on optimal action-value function, $Q^*(s, a)$, which is defined as the maximum of the expected cumulative discounted rewards by following any policy π, after taking action a in state s [2], given by:

$$Q^*(s, a) = \max_\pi \mathbb{E}\Big[R_{i+1} + \gamma R_{i+2} + \gamma^2 R_{i+3} + \ldots \;\Big|\; S_i = s, A_i = a; \pi \Big] \tag{8}$$

With Q^* being known, an optimal policy π^* is to greedily select the action with the highest value in any state.

To accommodate the single-ring RISC problem, our improved DDQN, termed Double Deep Dynamic Q-network (D3ynQN), has training settings different from the original ones in two aspects. First, we propose a new TD algorithm to accommodate the time-varying environment. Second, we introduce an operation to augment state transition samples. Algorithm 1 exhibits the pseudo-code of D3ynQN training process.

Algorithm 1: Training process of D3ynQN

Initialize the behavior network with random weights θ;
Initialize the target network with weights $\theta^- = \theta$;
Set the exploration probability $\epsilon = 1.0$;
Set the time step $i = 1$;
Set the state and action number $j = 1$;
`// see the hyper-parameters in Table 2`
while *have not selected enough actions* **do**
  Sample the vehicle volumes, turning ratios, and pedestrian volumes;
  Execute a simulation run to the moment of initial state;
  while *not reach the end of simulation period* **do**
    Select the j^{th} action A_i^j randomly with probability ϵ, otherwise $A_i^j = \text{argmax}_a Q(S_i^j, a; \theta)$;
    Execute the current simulation run to time step (i+d^j);
    Get the (j+1)th state $S_{i+d^j}^{j+1}$;
    Collect the cumulative discounted rewards RS^{j+1} between time steps i and (i+d) by Equation (11);
    Perform the data augmentation on transition $(S_i^j, A_i^j, RS^{j+1}, S_{i+d^j}^{j+1})$;
    Store the augmented transitions in replay buffer;
    Sample a mini-batch of transitions $(S_I^J, A_I^J, RS^{J+1}, S_{I+D^J}^{J+1})$ randomly from the replay buffer;
    Compute the mean TD error L^J for the mini-batch by Equation (15);
    Perform a SGD step on L^J with respect to θ;
    Reset $\theta^- = \theta$ every a certain number of SGD steps;
    Anneal ϵ if it is greater than the final value;
    Set $i = i + d^j$, $j = j + 1$;
  end while
  Stop the current simulation run;
end while

Table 2. Hyper-parameters of D3ynQN.

Hyper-Parameter	Value	Description
SGD Optimizer	Adam	For DDQN, Adam optimizer [49] was found to be less sensitive to the choice of learning rate [50].
γ	0.995	Empirically, the agent takes account of the future rewards within $1/(1-\gamma)$ time steps. γ = 0.995 leads to an effective time horizon of 200 s, which meets our expectation that the impact of an action is worth being observed in current and next one or two cycles.
Mini-batch	128	The behavior network's gradient is computed over 128 state transition samples. Basically, the larger the mini-batch, the smaller the variance of the gradient.
Replay Start	1500	Actions are randomly selected 1500 times before SGD starts. After symmetry augmentation we have 6000 state transition samples.
Replay Buffer	30,000	The replay buffer is full after 30,000 actions are selected. From then on, a new state transition sample replaces the oldest one. Numerically, *Replay Buffer* = 20 × *Replay Start*.
Final Epsilon	0.1 *	The value of ϵ in ϵ-greedy behavior policy is annealed to 0.1 eventually.
Annealing	30,000	The value of ϵ is linearly annealed over 30,000 actions selected. Numerically, *Annealing* = *Replay Buffer*.
Training Size	1,500,000	Training is done when 1,500,000 actions are selected. Numerically, *Training Size* = 50 × *Replay Buffer*. Note that SGD steps = *Training Size* - *Replay Start*.
Target Update	10,000 *	The target network is updated every 10,000 SGD steps.
Learning Rate	0.00025 *	SGD is performed based on the behavior network's gradient × 0.00025.

* These values are taken from those used in the original DDQN without tuning.

6.1. TD(Dyn)

6.1.1. Time-Step Size

Commonly, the time-step size of agent-environment interaction is defined as the interval between two successive actions. This interval may be not fixed throughout an RL

task, and thus varies the time-step size. The procedure of state observation, action selection, and reward collection is performed every time step.

Rewards are discounted as time step goes on. The discount rate γ represents the uncertainty for the agent about future rewards. Regarding the single-ring RISC problem, the uncertainty arises not only from the random nature of the agent's future actions, but from the time-varying nature of the environment. Therefore, the rewards should be discounted as time goes on, instead of only as more actions are selected.

To achieve this, we define the time-step size as one second throughout our RL task. The agent collects and discounts reward every time step, whereas observes state and selects action every a number of time steps.

6.1.2. TD Algorithm in DDQN

There is a series of n-step TD algorithms that can be used to update action values. The original DDQN is based on 1-step TD (also known as Q-Learning) algorithm, which updates the value $Q(S_i, A_i)$ of action A_i in state S_i, toward the TD target $R_{i+1} + \gamma \max_a Q(S_{i+1}, a)$, with:

$$Q(S_i, A_i) \leftarrow Q(S_i, A_i) + \alpha\big(R_{i+1} + \gamma \max_{a'} Q(S_{i+1}, a') - Q(S_i, A_i)\big) \tag{9}$$

where, S_i, A_i, R_i are the state, action, and reward at time step i, respectively; α is learning rate; and a' is bootstrap action.

By altering the number of time steps between an action and its bootstrap action, we can get different TD algorithms [2]. For instance, the TD target in 2-step TD algorithm is composed of the next two rewards and the estimated action value two time steps later: $R_{i+1} + \gamma R_{i+2} + \gamma^2 \max_{a'} Q(S_{i+2}, a')$.

6.1.3. Proposed TD Algorithm

According to our definition of time-step size, the number of time steps between two successive actions is not fixed. Consequently, the 1-step TD algorithm in DDQN is not applicable to solving the single-ring RISC problem.

The n-step TD algorithms suggest that, the TD target with an arbitrary number of next rewards is an estimator of the current action value. From this we present a new algorithm termed TD(Dyn). Based on the current action that the agent selects, TD(Dyn) dynamically adjusts the number of reward items in the TD target formula, so that bootstrapping will be performed just at the time step when the next action is selected.

We begin with d^j, which denotes the number of time steps between action A_i^j and its next action, given by:

$$d^j = A_i^j + YLW + CLR + G_{\min} \tag{10}$$

where A_i^j is the jth action at time step i; and YLW and CLR are the yellow change and red clearance intervals for the current vehicle phase, respectively. Note that we add the superscript "j" denoting the ordinal number of an action or state.

Now we know that the $(j+1)$th action is selected at time step $(i + d^j)$. Then we have cumulative discounted rewards RS^{j+1} between the jth and $(j+1)$th actions, given by:

$$RS^{j+1} = \sum_{t=i+1}^{i+d^j} \gamma^{t-i-1} R_t \tag{11}$$

Taking the $(j+1)$th action as the bootstrap action of the jth action, we get the TD target Y^j for the value of the jth action, given by:

$$Y^j = RS^{j+1} + \gamma^{(d^j)} \max_{a'} Q\left(S_{i+d^j}^{j+1}, a'\right) \tag{12}$$

where $S_{i+d^j}^{j+1}$ is the $(j+1)$th state at time step $(i+d^j)$.

Subsequently, we arrive at the TD(Dyn)'s formula of updating the action value $Q(S_i^j, A_i^j)$ toward Y^j, given by:

$$Q(S_i^j, A_i^j) \leftarrow Q(S_i^j, A_i^j) + \alpha\Big(Y^j - Q(S_i^j, A_i^j)\Big) \tag{13}$$

Based on TD(Dyn) algorithm above, the per-sample TD target in the original DDQN (decoupling action selection and evaluation with target network) is modified to:

$$Y^j_{\text{D3ynQN}} = RS^{j+1} + \gamma^{(d^j)} Q\Big(S^{j+1}_{i+d^j}, \text{argmax}_{a'} Q(S^{j+1}_{i+d^j}, a'; \theta); \theta^-\Big) \tag{14}$$

where θ and θ^- are the parameters of the behavior network and target network in DDQN, respectively.

Note that there is no terminal state for the single-ring RISC problem. Accordingly, Equation (14) gives no special treatment to the last state in a simulation run, which differs from that in the original DDQN.

Lastly, the per-sample TD error, i.e., the per-sample loss L^j of the behavior network for stochastic gradient descent (SGD), is given by:

$$L^j = \Big(Y^j_{\text{D3ynQN}} - Q(S_i^j, A_i^j; \theta)\Big)^2 \tag{15}$$

6.2. Symmetry Augmentation

AlphaGo augments its training data set by flipping and rotating the Go board for each board position [51]. Zheng et al. [37] leveraged the symmetry properties of intersection to design neural network. Inspired by these, we propose a data augmentation based on the symmetry between opposing intersection legs, which is specific to the single-ring RISC problem. It is worth noting that the data augmentation is an added bonus. Training can be done without it, but more effort need be invested to collect state transition samples $x^j = (S_i^j, A_i^j, RS^{j+1}, S^{j+1}_{i+d^j})$.

For a pair of opposing intersection legs, the symmetry augmentation is applicable if the following conditions are met:

(1) The opposing legs are the same in terms of the number of state matrics and the number of state features in each state matrix. In other words, they have identical road geometry and lane markings; and

(2) The change of the state features on one leg during the state transition from S_i^j to $S^{j+1}_{i+d^j}$ induced by action A_i^j, can also arise on the opposing leg. In other words, the opposing legs implement the leading or lagging left-turn phasing, so that the actions can lead to the release of opposing non-conflicting vehicle movements.

The way we organize state features by intersection leg offers convenience to the symmetry augmentation. We describe it as follows. For a pair of opposing legs, we simultaneously exchange the state features of their grids in both S_i^j and $S^{j+1}_{i+d^j}$, while keeping A_i^j and RS^{j+1} unchanged in the state transition sample.

For example, say the bth intersection leg and its opposing (b^-)th one both have two approach lane groups. S_i^j is given by:

$$\begin{aligned} S_i^j = \Big(&\ldots, MAT^{b,1}, MAT^{b,2}, MAT^{b}, \\ &\ldots, MAT^{b^-,1}, MAT^{b^-,2}, MAT^{b^-}, \ldots\Big) \end{aligned} \tag{16}$$

After exchanging the state features for the bth and (b^-)th intersection legs, we get $S_i^j|_b^{b^-}$ given by:

$$S_i^j|_b^{b^-} = \Big(\ldots, MAT^{b^-,1}, MAT^{b^-,2}, MAT^{b^-}, \ldots, MAT^{b,1}, MAT^{b,2}, MAT^{b}, \ldots\Big) \tag{17}$$

Note that the matrix items represented by the ellipses are not exchanged.

In the same way, we get $S_{i+d^j}^{j+1}|_b^{b^-}$. Then we form a new sample $x^j|_b^{b^-}$, given by:

$$x^j|_b^{b^-} = \left(S_i^j|_b^{b^-}, A_i^j, RS^{j+1}, S_{i+d^j}^{j+1}|_b^{b^-}\right) \tag{18}$$

In this study, each state transition sample is augmented to 4 symmetries by performing exchanging on two pairs of opposing intersection legs.

Obviously, the symmetry augmentation increases data utilization and shortens training time. In contrast to Zheng's way to leverage intersection symmetry [37], the additional advantage of our way lies in reducing the potential for the over-fitting of neural network, since not one but four samples are generated with the current behavior policy [52].

6.3. Hyper-Parameters

Table 2 shows the hyper-parameters of D3ynQN. Some of them are tuned, with reason, to accommodate the single-ring RISC problem.

7. Experiments

We conducted two experiments at a virtual intersection. The first was to verify whether the agent can improve the target policy toward better vehicle delay, in virtue of the proposed RL solution and the non-delay-based reward. The second was to compare the impact of the trained agent (TA) and a fully-actuated control technique (FACT) on the average vehicle delay.

The agent is trained at the virtual intersection modeled by the prevailing traffic simulation tool, Vissim 11. The agent interacts with the environment via Vissim COM interface for Python. The neural network is constructed by TensorFlow 1.15.

7.1. Virtual Intersection

Figure 4 shows the layout of the virtual intersection. The streets have three lanes in each direction of travel. There is a left-turn lane, two through lanes, and a shared through and right-turn lane on each intersection approach. Crosswalks are placed on all the intersection legs. The approach lanes are 50 m in length, and the approach tapers are 10 m in length. The posted speed limit is 50 km/h.

We simulate peak-period demand scenarios. The vehicle movements are composed of passenger cars only. For each intersection approach, the vehicle volume (unit in veh/h) is sampled from 1200, 1201, . . . , 1500. The left-turn and right-turn ratio by approach (unit in %) are sampled from 15.0, 15.1, . . . , 25.0 and 5.0, 5.1, . . . , 10.0, respectively. The unidirectional pedestrian volume on a crosswalk (unit in ped/h) is sampled from 100, 101, . . . , 150. The sampling work is performed before a simulation run starts. In doing so, the traffic demand varies between intersection approaches and between simulation runs.

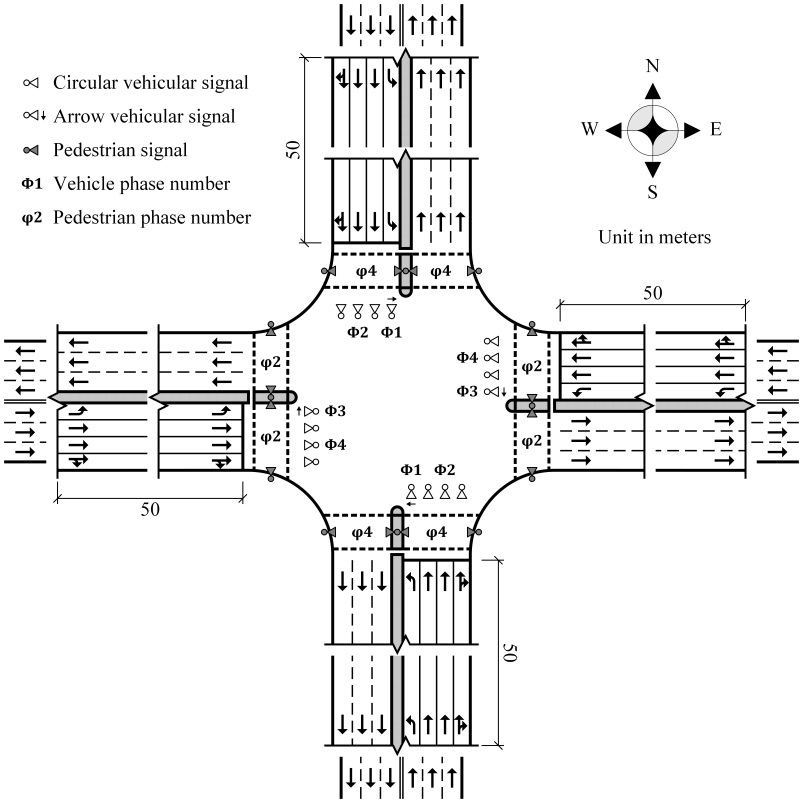

Figure 4. Layout of the virtual intersection.

We examine whether the peak-period demand scenarios are simulated successfully. As the experimental results exhibited below in Section 7.3 (experiment 2), the average vehicle delay at FACT-enabled intersections ranged from 33.6 s to 100.0 s over all 6000 simulation runs. According to Highway Capacity Manual 2010, the intersection operated at the level of service D or worse [53]. There is good reason to suppose that the intersection was in peak-period demand scenarios due to the traffic demand sampled.

The left-turn movements are protected and the right-turn movements are permitted. The right-turn vehicles must yield to pedestrians who are crossing the crosswalks. The leading left-turn phasing is used (see Figure 1a). The minimum greens are set to 15 s for the through phases and to 5 s for the left-turn phases. The maximum greens are 35 s greater than the minimum greens [42,54]. This leads to a total of 36 actions in the action space. For all the vehicle phases, the yellow change intervals are set to 3 s, and the red clearance intervals to separate vehicle-to-vehicle conflicts are set to 2 s.

The simulation period of a simulation run is 4200 simulation seconds (sim.s). The first 600 sim.s is the warm-up period. It means that an episode begins at 600 sim.s and is truncated at the length of 3600 sim.s. The purpose of setting the warm-up period is twofold. The first is to randomize the initial state of episode. The second is to load road users into the intersection and make it as busy as expected. The purpose of episode truncation is to increase the variety of state transition samples.

7.2. Experiment 1

Regarding DDQN, an increasing accuracy of the estimated action values is not a clear sign of policy improvement [52]. This is saying that TD error is incapable of reflecting training progress. On the other hand, the ϵ-greedy behavior policy randomly selects actions sometimes. Thus, the intersection performance or the cumulative rewards during training are mixed with the effect of exploring non-greedy actions, and cannot accurately track the training progress too.

We instead followed a periodic test procedure to measure the quality of target policy [4]. First, we fetched the behavior network every 50 episodes during training to conduct a test.

Second, we executed 50 simulation runs in a test with the target policy learned, which greedily selected the action with the maximal value for each state. Third, we collected the average vehicle delay at the intersection from 600 sim.s to 4200 sim.s in each simulation run, and computed the 15th, 50th, and 85th percentiles of the average vehicle delay for a test.

As shown in Figure 5, in the first 300,000 SGD steps, the 50th percentiles decreased, with fluctuation, from about 80 s to less than 50 s. It indicated notable improvement in the target policy, with the guide of the reward we defined. Since the 300,000th SGD step, the 50th percentiles fluctuated with no obvious upward or downward trend. Given the inherent instability of approximating Q^* by neural networks [4,52], there were still opportunities to improve the target policy in the remaining SGD steps. Indeed, the best 50th percentile, i.e., 40.52 s, came at the 1,105,796th SGD step. We saved the behavior network parameters at that time to generate the target policy finally learned [52], which was used to conduct experiment 2.

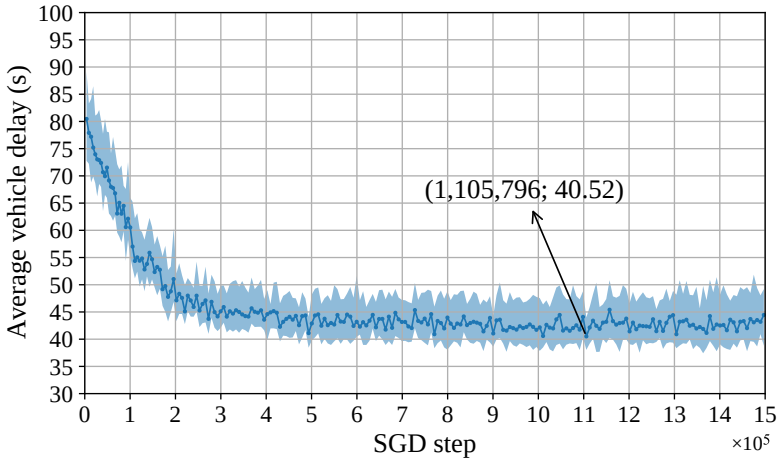

Figure 5. Training curve of the agent. Solid line represents the 50th percentiles of the average vehicle delay across the 50 simulation runs in each test, and shaded area covers the 15–85th percentiles.

7.3. Experiment 2

FACT uses logic rules to solve the single-ring RISC problem. Its goal is to finitely extend current vehicle phase to serve its continued demand. As required by RiLSA [54], a vehicle detector is placed on each approach lane, dp meters upstream of the stop line, to detect time headways. The current vehicle phase ends if either of the following conditions is met: (1) all the detectors of the phase have detected a headway greater than gt seconds after the expiration of its minimum green; and (2) the phase reaches its maximum green.

gt is known as the "gap time". RiLSA requires the value of gt to be between 2 s and 3 s during peak periods. In this study, we conduct the experiment with gt being set to 2.0 s, 2.5 s, and 3.0 s, respectively. In addition, RiLSA requires the value of dp to be set to the product of gt and the posted speed limit. This is to allow the last vehicle triggering the detector to approach the stop line before the end of green [54].

With the lane-by-lane detectors, FACT essentially employs the time-series data of each vehicle's headway. Actually, FACT constructs a point-based "state representation" in a sense. Compared with the grid-based representation in D3ynQN, FACT in a way also fully exploits the information of every vehicle.

Given that the geometric design and the phase sequence are fixed over time, the average vehicle delay is governed by the traffic demand and the signal operations. So we controlled the building elements of the traffic demand to investigate the impacts of TA and FACT on the average vehicle delay. The controlling factors and their levels are given as follows.

1. The vehicle volume on each intersection approach (veh/h): 1200, 1210, . . . , 1500.

2. The left-turn ratio on each intersection approach (%): 15.0, 15.2, . . . , 25.0.
3. The right-turn ratio on each intersection approach (%): 5.0, 5.2, . . . , 10.0.
4. The unidirectional pedestrian volume on each crosswalk (ped/h): 100, 110, . . . , 150.

Considering the number and levels of controlling factors, we performed a D-optimal design [55] by using Ngene 1.1.1. For each value of gap time, a total of 2000 demand scenarios were generated for the intersection. The D-error was 1.3×10^{-5}. For each demand scenario, we executed four simulation runs, with the signals being timed by TA, and by FACT with different gap times, respectively. The average vehicle delay was measured from 600 sim.s to 4200 sim.s. The distributions of the data measured are presented in box plots.

As shown in Figure 6, the 25th, 50th, and 75th percentiles of the average vehicle delay were unexceptionally lower at the TA-enabled intersection than at the FACT-enabled ones, regardless of the gap times. It implied that TA allowed the intersection to operate with generally better performance. On the other hand, the differences in the inter-quartile ranges and whisker ranges indicated that more stable performance could be achieved at the TA-enabled intersection.

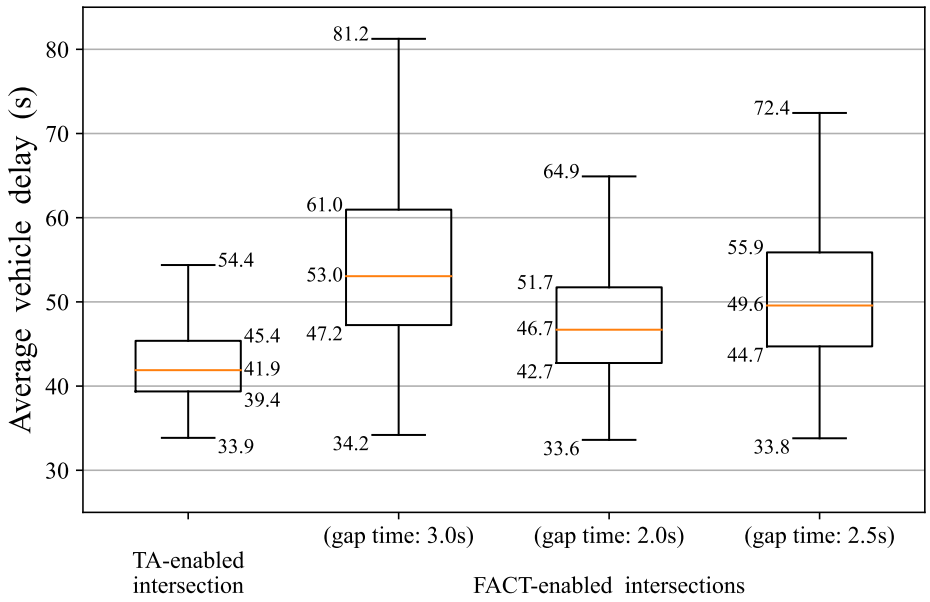

Figure 6. Box plots of the average vehicle delay at the TA-enabled and FACT-enabled intersections with different gap times.

We draw a scatter plot to investigate the average vehicle delay at the TA-enabled and FACT-enabled intersections in each demand scenario. As shown in Figure 7, the average vehicle delay was smaller at the TA-enabled intersection than at the FACT-enabled one in the vast majority of demand scenarios with any gap time. The advantage of the TA-enabled intersection could be 10 s or greater, whereas the disadvantage of the TA-enabled intersection was ignorable.

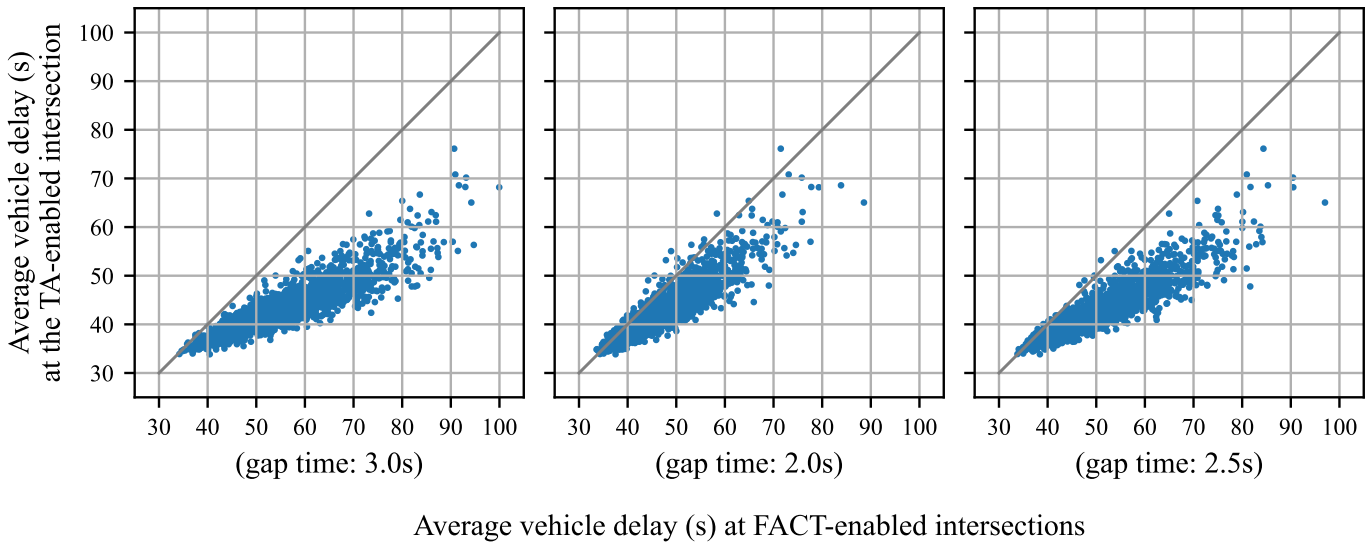

Figure 7. Scatter plot of the average vehicle delay at the TA-enabled and FACT-enabled intersections in each demand scenario with different gap times. Each dot represents a demand scenario, with the x-coordinate and y-coordinate denoting the average vehicle delay at the FACT-enabled and TA-enabled intersections, respectively.

We conducted a paired samples *t*-test at the significance level of 0.05 to investigate the average vehicle delay at the TA-enabled and FACT-enabled intersections. As shown in Table 3, the mean of the average vehicle delay was 42.98 s at the TA-enabled intersection. Intuitively, it was 4.82 s or 10.1% lower than that of 47.81 s at the FACT-enabled intersection even with the best performing gap time. Such a performance advantage could increase to 11.95 s or 21.8% in the most favorable case for TA. Statistically, there was a significant difference in the average vehicle delay, whatever the value of gap time. The corresponding Cohen's *d* was greater than 1.2, meaning that such a significant difference had a very large effect size [56].

Table 3. Paired samples *t*-test on the average vehicle delay at the TA-enabled and FACT-enabled intersections with different gap times.

Item	**Content**		
Gap time in FACT	3.0 s	2.0 s	2.5 s
Mean difference	−11.95 (42.98 [a]–54.93 [b])	−4.82 (42.98 [a]–47.81 [b])	−8.16 (42.98 [a]–51.14 [b])
Std. deviation	6.44	3.42	4.93
Std. error of mean	0.14	0.08	0.11
95% confidence interval	[−12.23, −11.67]	[−4.97, −4.67]	[−8.37, −7.94]
t	−82.936	−63.082	−73.968
Sig. (2-tailed)	0.000 **	0.000 **	0.000 **
Cohen's *d*	1.855	1.411	1.654

[a] TA-enabled intersection. [b] FACT-enabled intersection. ** at 0.05 significance level.

8. Conclusions

This study provides an RL solution called "D3ynQN" for the single-ring RISC problem from the perspective of traffic engineers. The key to success lies in using traffic engineering expertise to handpick an RL method and improve it in terms of RL elements and training settings.

There are four academic contributions in this study:

1. We analyze the applicability of various categories of RL methods to the single-ring RISC problem. We found that such problem can be well solved by value-based off-policy deep RL methods.
2. We argue that the reward defined with vehicle departures is training-friendly, as the short-term rewards can properly encourage and punish the agent's behavior. We experimentally demonstrate that the agent can learn from such non-delay-based reward to gradually lower the average vehicle delay for the single-ring RISC problem.
3. We argue that the rewards should be discounted with time in the time-varying traffic environments. Accordingly, we define the time-step size as one second, and present an action-value updating algorithm TD(Dyn) to match the varying number of time steps between actions.
4. We propose a data augmentation algorithm based on intersection symmetry as an added bonus for training.

In addition, this study has three practical contributions:

1. The proposed RL solution is subject to the signal timing constraints in engineering, which include fixed phase sequence, minimum and maximum greens, yellow change and red clearance intervals.
2. The proposed RL solution is validated at a closer-to-reality intersection with varying vehicle turning ratios and pedestrian demand.
3. With the proposed RL solution, the trained agent leads to much smaller average vehicle delay than a properly configured fully-actuated control technique in a statistical sense.

9. Discussion

From the perspective of traffic engineers, one of the concerns is whether an RL-based technique can be applied in practice.

We believe the proposed RL solution "D3ynQN" holds the potential to be implemented at real intersections. First, the required state and reward data are available in practice, as we point out in Sections 5.1.3 and 5.3.2. This enables the RL loop of "getting states—taking actions—receiving rewards—improving the policy" to be performed. Second, our solution abides by the signal timing constraints in engineering. It means the basic safety requirements can be satisfied.

However, there are still many long-standing challenges facing the successful application of RL-based signal control in practice. For instance, (1) the generalization of the knowledge learned at an intersection to other intersections; (2) the gap between virtual and real-world intersections' dynamics.

Author Contributions: Conceptualization, Q.Z. and H.X.; Formal analysis, Q.Z. and D.Z.; Funding acquisition, H.X. and J.C.; Investigation, H.X. and G.T.; Methodology, Q.Z.; Project administration, Q.Z. and K.Z.; Resources, H.X., J.C. and G.T.; Software, H.X. and D.Z.; Supervision, H.X. and J.C.; Validation, Q.Z. and H.X.; Visualization, Q.Z. and K.Z.; Writing—original draft, Q.Z.; Writing—review & editing, Q.Z. and H.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (grant number: 61374193) and the Humanities and Social Science Foundation of Ministry of Education of China (grant number: 19YJCZH201).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RISC	real-time isolated signal control
RL	reinforcement learning
DDQN	double deep Q-network
DQN	deep Q-network
DNNs	deep neural networks
ReLU	Rectifier Linear Units
TD	temporal-difference
SGD	stochastic gradient descent
TA	trained agent
FACT	fully-actuated control technique

References

1. Bucolo, M.; Buscarino, A.; Famoso, C.; Fortuna, L. Chaos addresses energy in networks of electrical oscillators. *IEEE Access* **2021**, *9*, 153258–153265. [CrossRef]
2. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
3. Bingham, E. Reinforcement learning in neurofuzzy traffic signal control. *Eur. J. Oper. Res.* **2001**, *131*, 232–241. [CrossRef]
4. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
5. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 750–797. [CrossRef]
6. Canese, L.; Cardarilli, G.C.; Di Nunzio, L.; Fazzolari, R.; Giardino, D.; Re, M.; Spanò, S. Multi-agent reinforcement learning: A review of challenges and applications. *Appl. Sci.* **2021**, *11*, 4948. [CrossRef]
7. Ma, D.; Zhou, B.; Song, X.; Dai, H. A deep reinforcement learning approach to traffic signal control with temporal traffic pattern mining. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 11789–11800. [CrossRef]
8. Yang, X.; Xu, Y.; Kuang, L.; Wang, Z.; Gao, H.; Wang, X. An information fusion approach to intelligent traffic signal control using the joint methods of multiagent reinforcement learning and artificial intelligence of things. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 9335–9345. [CrossRef]
9. Alegre, L.N.; Ziemke, T.; Bazzan, A.L.C. Using reinforcement learning to control traffic signals in a real-world scenario: An approach based on linear function approximation. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 9126–9135. [CrossRef]
10. Jin, J.; Ma, X. Hierarchical multi-agent control of traffic lights based on collective learning. *Eng. Appl. Artif. Intell.* **2018**, *68*, 236–248. [CrossRef]
11. Xiong, Y.; Zheng, G.; Xu, K.; Li, Z. Learning traffic signal control from demonstrations. In Proceedings of the CIKM '19: 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 2289–2292.
12. Rizzo, S.G.; Vantini, G.; Chawla, S. Time critic policy gradient methods for traffic signal control in complex and congested scenarios. In Proceedings of the KDD '19: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1654–1664.
13. Mo, Z.; Li, W.; Fu, Y.; Ruan, K.; Di, X. CVLight: Decentralized learning for adaptive traffic signal control with connected vehicles. *Transp. Res. Pt. C-Emerg. Technol.* **2022**, *141*, 103728. [CrossRef]
14. Zhu, H.; Wang, Z.; Yang, F.; Zhou, Y.; Luo, X. Intelligent traffic network control in the era of internet of vehicles. *IEEE Trans. Veh. Technol.* **2021**, *70*, 9787–9802. [CrossRef]
15. Yang, S.; Yang, B.; Kang, Z.; Deng, L. IHG-MA: Inductive heterogeneous graph multi-agent reinforcement learning for multi-intersection traffic signal control. *Neural Netw.* **2021**, *139*, 265–277. [CrossRef]
16. Li, Z.; Yu, H.; Zhang, G.; Dong, S.; Xu, C.Z. Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning. *Transp. Res. Pt. C-Emerg. Technol.* **2021**, *125*, 103059. [CrossRef]
17. Liu, J.; Qin, S.; Luo, Y.; Wang, Y.; Yang, S. Intelligent traffic light control by exploring strategies in an optimised space of deep Q-learning. *IEEE Trans. Veh. Technol.* **2022**, *71*, 5960–5970. [CrossRef]
18. Gu, J.; Lee, M.; Jun, C.; Han, Y.; Kim, Y.; Kim, J. Traffic signal optimization for multiple intersections based on reinforcement learning. *Appl. Sci.* **2021**, *11*, 10688. [CrossRef]
19. Zhang, R.; Ishikawa, A.; Wang, W.; Striner, B.; Tonguz, O.K. Using reinforcement learning with partial vehicle detection for intelligent traffic signal control. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 404–415. [CrossRef]

20. Xu, K.; Huang, J.; Kong, L.; Yu, J.; Chen, G. PV-TSC: Learning to control traffic signals for pedestrian and vehicle traffic in 6G era. *IEEE Trans. Intell. Transp. Syst.* **2022**, 1–12. Available online: https://ieeexplore.ieee.org/document/9733963 (accessed on 27 July 2022).
21. Mao, F.; Li, Z.; Li, L. A comparison of deep reinforcement learning models for isolated traffic signal control. *IEEE Intell. Transp. Syst. Mag.* **2022**, 2–22. [CrossRef]
22. Joo, H.; Lim, Y. Traffic signal time optimization based on deep Q-network. *Appl. Sci.* **2021**, *11*, 9850. [CrossRef]
23. Yoon, J.; Ahn, K.; Park, J.; Yeo, H. Transferable traffic signal control: Reinforcement learning with graph centric state representation. *Transp. Res. Pt. C-Emerg. Technol.* **2021**, *130*, 103321. [CrossRef]
24. Xiao, N.; Yu, L.; Yu, J.; Chen, P.; Liu, Y. A cold-start-free reinforcement learning approach for traffic signal control. *J. Intell. Transport. Syst.* **2022**, *26*, 476–485. [CrossRef]
25. Devailly, F.X.; Larocque, D.; Charlin, L. IG-RL: Inductive graph reinforcement learning for massive-scale traffic signal control. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 7496–7507. [CrossRef]
26. Chu, K.F.; Lam, A.Y.S.; Li, V.O.K. Traffic signal control using end-to-end off-policy deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 7184–7195. [CrossRef]
27. Tan, T.; Bao, F.; Deng, Y.; Jin, A.; Dai, Q.; Wang, J. Cooperative deep reinforcement learning for large-scale traffic grid signal control. *IEEE Trans. Cybern.* **2020**, *50*, 2687–2700. [CrossRef]
28. Gu, J.; Fang, Y.; Sheng, Z.; Wen, P. Double deep Q-network with a dual-agent for traffic signal control. *Appl. Sci.* **2020**, *10*, 1622. [CrossRef]
29. Lee, J.; Chung, J.; Sohn, K. Reinforcement learning for joint control of traffic signals in a transportation network. *IEEE Trans. Veh. Technol.* **2020**, *69*, 1375–1387. [CrossRef]
30. Liang, X.; Du, X.; Wang, G.; Han, Z. A deep reinforcement learning network for traffic light cycle control. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1243–1253. [CrossRef]
31. Wei, H.; Chen, C.; Zheng, G.; Wu, K.; Gayah, V.; Xu, K.; Li, Z. PressLight: Learning max pressure control to coordinate traffic signals in arterial network. In Proceedings of the KDD '19: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1290–1298.
32. Zang, X.; Yao, H.; Zheng, G.; Xu, N.; Xu, K.; Li, Z. MetaLight: Value-based meta-reinforcement learning for traffic signal control. In Proceedings of the 34th AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 1153–1160.
33. Boukerche, A.; Zhong, D.; Sun, P. A novel reinforcement learning-based cooperative traffic signal system through max-pressure control. *IEEE Trans. Veh. Technol.* **2022**, *71*, 1187–1198. [CrossRef]
34. Joo, H.; Lim, Y. Intelligent traffic signal phase distribution system using deep Q-network. *Appl. Sci.* **2022**, *12*, 425. [CrossRef]
35. Wang, H.; Yuan, Y.; Yang, X.T.; Zhao, T.; Liu, Y. Deep Q learning-based traffic signal control algorithms: Model development and evaluation with field data. *J. Intell. Transport. Syst.* **2022**, 1–21. Available online: https://www.tandfonline.com/doi/full/10.1080/15472450.2021.2023016 (accessed on 27 July 2022).
36. Genders, W.; Razavi, S. Asynchronous n-step Q-learning adaptive traffic signal control. *J. Intell. Transport. Syst.* **2019**, *23*, 319–331. [CrossRef]
37. Zheng, G.; Xiong, Y.; Zang, X.; Feng, J.; Wei, H.; Zhang, H.; Li, Y.; Xu, K.; Li, Z. Learning phase competition for traffic signal control. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1963–1972.
38. Wei, H.; Zheng, G.; Yao, H.; Li, Z. IntelliLight: A reinforcement learning approach for intelligent traffic light control. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2496–2505.
39. Zhang, H.; Liu, C.; Zhang, W.; Zheng, G.; Yu, Y. GeneraLight: Improving environment generalization of traffic signal control via meta reinforcement learning. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, Virtual, Ireland, 19–23 October 2020; pp. 1783–1792.
40. Wang, X.; Yin, Y.; Feng, Y.; Liu, H.X. Learning the max pressure control for urban traffic networks considering the phase switching loss. *Transp. Res. Pt. C-Emerg. Technol.* **2022**, *140*, 103670. [CrossRef]
41. Aslani, M.; Mesgari, M.S.; Wiering, M. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transp. Res. Pt. C-Emerg. Technol.* **2017**, *85*, 732–752. [CrossRef]
42. Urbanik, T.; Tanaka, A.; Lozner, B.; Urbanik, T.; Tanaka, A.; Lozner, B.; Lindstrom, E.; Lee, K.; Quayle, S.; Beaird, S.; et al. *NCHRP Report 812: Signal Timing Manual*, 2nd ed.; Transportation Research Board: Washington, DC, USA, 2015.
43. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
44. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
45. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. In Proceedings of the 4th International Conference Learning Representations, San Juan, PR, USA, 2–4 May 2016.
46. van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-learning. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 2094–2100.

47. Papageorgiou, M.; Kotsialos, A. Freeway ramp metering: An overview. *IEEE Trans. Intell. Transp. Syst.* **2002**, *3*, 271–281. [CrossRef]
48. Nair, V.; Hinton, G.E. Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 807–814.
49. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference Learning Representations, San Diego, CA, USA, 7–9 May 2015.
50. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the 32th AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 3215–3222.
51. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Hassabis, D. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
52. Roderick, M.; MacGlashan, J.; Tellex, S. Implementing the deep Q-network. In Proceedings of the 30th Conference Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
53. TRB Committee on Highway Capacity and Quality of Service. *HCM2010: Highway Capacity Manual*, 5th ed.; Transportation Research Board: Washington, DC, USA, 2010.
54. Steering Committee Traffic Control and Traffic Safety. *Guidelines for Traffic Signals (RiLSA)*; Road and Transportation Research Association (FGSV): Cologne, Germany, 2003. (In English)
55. Vanniyasingam, T.; Daly, C.; Jin, X.; Zhang, Y.; Foster, G.; Cunningham, C.; Thabane, L. Investigating the impact of design characteristics on statistical efficiency within discrete choice experiments: A systematic survey. *Contemp. Clin. Trials Commun.* **2018**, *10*, 17–28. [CrossRef] [PubMed]
56. Sawilowsky, S.S. New effect size rules of thumb. *J. Mod. Appl. Stat. Meth.* **2009**, *8*, 26. [CrossRef]