Transportation Research Part C 141 (2022) 103728

Contents lists available at ScienceDirect

Transportation Research Part C

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

CVLight: Decentralized learning for adaptive traffic signal control with connected vehicles

Zhaobin Mo [b,1], Wangzhi Li [a,1], Yongjie Fu [b], Kangrui Ruan [b], Xuan Di [a,b,*]

[a] Data Science Institute, Columbia University, United States of America
[b] Department of Civil Engineering and Engineering Mechanics, Columbia University, United States of America

ARTICLE INFO

Keywords:
Traffic signal control
Deep reinforcement learning
Connected vehicles
Actor–critic algorithm

ABSTRACT

This paper develops a decentralized reinforcement learning (RL) scheme for multi-intersection adaptive traffic signal control (TSC), called "CVLight", that leverages data collected from connected vehicles (CVs). The state and reward design facilitates coordination among agents and considers travel delays collected by CVs. A novel algorithm, Asymmetric Advantage Actor-critic (Asym-A2C), is proposed where both CV and non-CV information is used to train the critic network, while only CV information is used to execute optimal signal timing. Comprehensive experiments show the superiority of CVLight over state-of-the-art algorithms under a 2-by-2 synthetic road network with various traffic demand patterns and penetration rates. The learned policy is then visualized to further demonstrate the advantage of Asym-A2C. A pre-train technique is applied to improve the scalability of CVLight, which significantly shortens the training time and shows the advantage in performance under a 5-by-5 road network. A case study is performed on a 2-by-2 road network located in State College, Pennsylvania, USA, to further demonstrate the effectiveness of the proposed algorithm under real-world scenarios. Compared to other baseline models, the trained CVLight agent can efficiently control multiple intersections solely based on CV data and achieve the best performance, especially under low CV penetration rates.

1. Introduction

Connected vehicle (CV) refers to the vehicular technology that enables vehicle-to-vehicle (V2V) (or vehicular ad-hoc networks (VANETs) and vehicle-to-infrastructure (V2I) communication (USDOT, 2019; Di and Shi, 2021). The enabling technology includes dedicated short-range communication (DSRC) (Abboud et al., 2016), or cellular communication like 5G (Agiwal et al., 2016; Chettri and Bera, 2020; Khurpade et al., 2018). Connected vehicles will likely generate terabytes of streaming data daily (SAS, 2015), holding great potential for various transportation applications including traffic signal control (TSC), which is the focus of this paper.

1.1. Literature review

CV can provide more detailed traffic information, real-time vehicle trajectories for example, when compared to traditional detectors. To incorporate CV data into TSC systems, researchers have developed various TSC strategies in recent years (Liu et al., 2014, 2017; Li et al., 2016; Li and Ban, 2018; Kim et al., 2019; Hussain et al., 2020; Li and Ban, 2020; Yan et al.). Although these

* Corresponding author at: Department of Civil Engineering and Engineering Mechanics, Columbia University, United States of America.
E-mail address: sharon.di@columbia.edu (X. Di).
[1] Zhaobin Mo and Wangzhi Li contributed equally to this work.

https://doi.org/10.1016/j.trc.2022.103728
Received 1 April 2021; Received in revised form 22 February 2022; Accepted 9 May 2022
Available online 1 June 2022

Major Notations

Reinforcement Learning

$\mathbf{s}$	State
S	State space
a	Action
A	Action space
o	Observation
O	Observation space
r	Reward
R	Reward space
Pr	State transition function
γ	Discounted factor
π	Policy
$Q(s,a)$	State–action value function
$V(s)$	State value function
θ	Parameters of critic's value neural network
ϕ	Parameters of actor's policy neural network

Traffic Signal Control

I	Set of all intersections
$L_{in}(i)$	Set of incoming lanes of intersection i
$L_{out}(i)$	Set of outgoing lanes of intersection i
p_i	Current phase of intersection i
d_i	Current phase duration of intersection i
$n_i(l)$	Number of vehicles in lane l at intersection i
$x_i(l)$	Scaled average delay per vehicle of lane l at intersection i
$t_{min}(l)$	Expected minimum travel time of lane l
D_i	Cumulative delay of all vehicles at intersection i
P_i	Pressure of intersection i

proposed TSC strategies with CV data significantly outperform traditional traffic signal control methods such as actuated TSC or fixed-time TSC, usually a 100% CV market penetration rate is assumed. Such a full market may not be achievable in the short term, thus motivating researchers to develop TSC strategies capable of running on limited CV data, specifically for low CV market penetration (below 30%) scenarios. In this section, we will mainly focus on TSC strategies that assume only a portion of vehicles are CVs and categorize them into **non-learning based** and **reinforcement learning (RL) based** TSC strategies. Readers who are interested in a more comprehensive overview of traffic control with connected vehicles can refer to Guo et al. (2019).

1.1.1. Non-learning based TSC with partial CV information

One major non-learning based TSC strategy includes formulating the TSC problem as an optimization problem (Li et al., 2013). At each time step, a TSC system receives environmental inputs (such as new vehicle arrivals) and updates traffic states (such as queue lengths) that can be partially or fully observable. The objective of the TSC system is to optimize some performance measurements (such as total vehicle delays) over a finite period of time. The output is a series of control signals to the TSC system that are usually represented by traffic signal phase sequences and durations. The most common optimization methods for TSC problems are mixed-integer linear programming (MILP) (He et al., 2012) and dynamic programming (DP) (Feng et al., 2015, 2016, 2018; Hu et al., 2019). Beak et al. (2017) further develop a 2-level optimization framework: at the intersection-level a DP is used to optimize the individual vehicle delays, and at the corridor level an MILP is solved to minimize platoon delays. Greedy rules like a longest-queue-first algorithm are also adopted to solve the TSC problem (Lee et al., 2013; Goodall et al., 2013).

To provide accurate state variables and objective measurements under low CV penetration rates, various traffic state estimation methods have been developed. They are categorized into *vehicle-level* and *flow-level* estimation methods based on information granularity. Vehicle-level algorithms estimate trajectories of each vehicle, while flow-level ones focus on aggregate traffic measurements such as traffic volumes.

For vehicle-level estimation, a majority of studies use CVs to infer positions and speeds of non-CVs (Goodall et al., 2014; Feng et al., 2015, 2016; Beak et al., 2017). Goodall et al. (2013) propose a predictive microscopic simulation algorithm (PMSA), which optimizes traffic signal timing based on simulated CVs over a time horizon from a microscopic numerical simulator. Without

inference of non-CVs, the proposed TSC system (PMSA) requires at least a 50% penetration rate to outperform a conventional TSC system. Goodall et al. (2014) further estimate unconnected vehicles (denoted by non-CVs) positions to improve the performance of PMSA under low penetration rates: stopping non-CVs are inserted into detected gaps among CVs and added to simulation with CVs. Feng et al. (2015, 2016) first divide road segments near intersections into three regions based on vehicle status, namely, a queuing region, a slow-down region, and a free-flow region. Then locations and speeds of individual non-CVs are estimated within each region.

Flow-level estimation infers queue lengths (Priemer and Friedrich, 2009; Tiaprasert et al., 2015; Li et al., 2020), platoon arrival time (He et al., 2012), vehicle densities (Mohebifard and Hajbabaie, 2018; Mohebifard et al., 2019; Al Islam et al., 2020), cumulative travel time (Lee et al., 2013), or traffic volumes (Zheng and Liu, 2017; Feng et al., 2018). Specifically, queue estimation for signalized intersections has gained more attention from researchers in recent years (Tiaprasert et al., 2015; Hao and Ban, 2015; Yang and Menendez, 2018; Gao et al., 2019; Li et al., 2020), especially by utilizing vehicle trajectory data (Hao and Ban, 2015; Yang and Menendez, 2018). Most recently, some studies focus on extremely low (below 10%) penetration rate scenarios. Feng et al. (2018) apply a traffic volume estimation method (Zheng and Liu, 2017) to an adaptive TSC system developed in Feng et al. (2015). This traffic volume estimation method utilizes trajectory data from CVs or navigation devices to estimate current traffic volumes under a penetration rate as low as 10%. To cope with low penetration rate scenarios, Al Islam et al. (2020) propose two kinds of traffic state estimation algorithms: one uses combined data from CVs and loop detectors to estimate the non-CV trajectories based on a car-following model, the other converts vehicle detection time to spatial distributions of CVs and non-CVs. Results of experiments on a real-world corridor of 4 intersections with real-world traffic demands show that both estimation methods function well under 10% or lower penetration rates.

Table 1 summarizes recent studies on non-learning based TSC with partial observation (i.e. without a 100% CV penetration rate). Each row represents the reference reviewed above, containing details of their information source, environment, estimation granularity, estimation methods, benchmarks, required minimum penetration rate, key assumptions, and gap, respectively.

In summary, when it comes to using CV data to estimate traffic states, non-learning based TSC methods usually rely on traffic models or certain assumptions of traffic. Because the performance of such TSC systems highly depends on the accuracy of traffic state estimation from CV data, the reliability of the adopted traffic models and assumptions are thus crucial.

1.1.2. Reinforcement Learning (RL) based TSC with partial CV information

One major difference between non-learning based and RL based strategies lies in their dependency on traffic models. RL based methods do not rely on traffic models, and instead, they learn from past experience and adjust agents' behavior to local environments.

Recently, RL based multi-intersection TSC problem gains more attention from researchers (Chu et al., 2019; Chacha Chen et al., 2020; Gong et al., 2019; Yang et al., 2019). Despite an increasing number of papers on RL based TSC using CV data published in recent years (Kim et al., 2019; Hussain et al., 2020; Yan et al.; Liu et al., 2014, 2017), only a few pay attention to the performance of proposed RL based TSC systems under low CV penetration rate scenarios, especially in large road networks. Aziz et al. (2019) evaluate their previous RL based TSC method (Al Islam et al., 2018) under various penetration rates in two real-world road networks. Wu et al. (2020) also test the performance of their multi-agent RL based TSC algorithm under different penetration rates. In Zhang et al. (2020), a series of more comprehensive experiments on the proposed RL based TSC method under different penetration rates and traffic demand patterns are conducted, but those experiments are all limited in an isolated intersection. Key features of these studies as well as the proposed method of this paper are summarized in Table 2, including their traffic information source, environment, agent, algorithm, benchmarks, state, action, reward, and gap, respectively.

Aziz et al. (2019) find that their RL based TSC method cannot learn well under penetration rates below 40%; Zhang et al. (2020) and Wu et al. (2020) show the robustness of proposed RL based TSC methods against low penetration rates. For instance, the RL based TSC system in Zhang et al. (2020) leads to an 80% decrease in waiting time at the 20% penetration rate, compared to its performance at 100% penetration rate. The difference in the performance of these RL based TSC algorithms at low penetration rates can be explained by the design of RL based TSC systems and experiments. As mentioned before, Zhang et al. (2020) use non-CV delay as part of rewards during training, but (Aziz et al., 2019) did not. Instead, CV data in Aziz et al. (2019) is utilized to estimate queue lengths in a simplified way: the distance from the position of the last stopping CV to the stop line is assumed to be the queue length of that lane. In addition, RL based TSC agents in Aziz et al. (2019) are trained at the 100% penetration rate but tested under various penetration rates, which are not able to generalize well. As to Wu et al. (2020), a recurrent neural network (RNN) is applied, which can learn historical information from time-continuous traffic state data. Experiment results show that the system with an RNN layer is more robust against scenarios with partial observation available (20% penetration rate or higher) than the same algorithm without such layer.

Previous studies in RL based TSC under various CV penetration rates demonstrate the great potential of RL based methods. However, an RL based TSC method using CV data and specifically designed for scenarios with low penetration rates and multiple intersections is missing. It remains unclear what CV information can improve robustness against partial observability. Also, the RL structure suitable for partially observed systems remains to be explored.

1.2. Contributions of this paper

To mitigate the limits we mentioned in the above section, this paper proposes a reinforcement learning scheme, CVLight, for the multi-intersection TSC problem without a full CV penetration rate. To investigate what CV information can improve robustness against partial observation, we incorporate vehicle delays and phase durations into the state design. By exploring the structure of the actor-critic algorithm, we include the information from both CVs and non-CVs into training.

Table 1
Existing research of non-learning based TSC with low penetration rates.

References	Information source	Environment	Estimation granularity	Estimation methods	Benchmarks	Required minimum penetration rate[a]	Key assumptions	Gap
Priemer and Friedrich (2009)	CV data; Loop detectors	A real-world network of 9 intersections with real-world traffic data	Flow-level	Estimating queue length based on current and historical CV data	Well-tuned fixed-time TSC	33%	No constraints on the number of phases, phase transitions, sequences or timings	Non-learning based methods usually rely on certain assumptions or traffic models for non-CV states estimation
He et al. (2012)	CV data	A real-world corridor of 8 intersections	Flow-level	Identifying platoons by analyzing headways of CVs; estimating platoon parameters using a linear regression model.	Well-tuned fixed-time TSC	40%	Penetration rate is given; Passenger vehicles constitute a significant majority of the vehicles in the network	
Lee et al. (2013)	CV data; Loop detectors or camera	A synthetic isolated intersection	Flow-level	Applying Kalman filtering to estimate the cumulative travel times	Actuated TSC	30%	Total vehicle counts are available	
Goodall et al. (2013)	CV data	A real-world corridor of 4 intersections	Flow-level	Populating the CV data into a microscopic traffic simulation to measure objective function from the simulated behavior	Actuated TSC	50%	The turning movement of vehicles are based on their current lanes	
Goodall et al. (2014)	CV data	A real-world corridor of 4 intersections	Vehicle-level	Estimating the positions of non-CVs based on the gaps among stopping CVs	The same TSC without non-CV estimation	10%–25%	Vehicles behaviors follow the Wiedemann car-following model; Length of CVs and gap between vehicles are known	
Tiaprasert et al. (2015)	CV data	A synthetic isolated intersection	Flow-level	Estimating queue length based on stopping and moving CV data.	Fixed-time TSC; actuated TSC	Not Available	Penetration rate is given; The individual location and speed of connected vehicle can be collected	
Feng et al. (2015, 2016)	CV data	A real-world isolated intersection	Vehicle-level	Using CV data to estimate speeds and locations of non-CVs	Actuated TSC	25% (Feng et al., 2015), $\leq$ 10% (Feng et al., 2016)	Road segments of traffic movements can be divided into three regions;	
Beak et al. (2017)	CV data	A synthetic corridor of 5 intersections	Vehicle-level	Using CV data to estimate speeds and locations of non-CVs	Actuated TSC	25%	Road segments of traffic movements can be divided into three regions	
Feng et al. (2018)	CV data	A real-world isolated intersection	Flow-level	Estimating cycle-by-cycle vehicle arrival times and delays based on estimated average historical volume and a limited number of observed critical CV trajectories	Actuated TSC	<10%	Vehicle arrivals follow Poisson process; Vehicle length is uniform and known: Free travel times are the same for all vehicles	
Mohebifard et al. (2019)	CV data; Loop detectors; Infrastructure to infrastructure communications	A real-world network of 20 intersections	Flow-level	Estimating density across network links using data from CVs and loop detectors	The solution to the central problem	30%	The dynamic travel demand is known; vehicle counts and stop bar detectors are available at all intersections	
Al Islam et al. (2020)	CV data; Loop detectors; Infrastructure to infrastructure communications	A real-world corridor of 4 intersections with real-world traffic data	Flow-level	(1) Integrating data from CVs and loop detectors to estimate the trajectories of non-CVs based on car-following concepts. (2) Converting the temporal point vehicle detections to a spatial vehicle distribution on a link.	Real-world signal plan in the case study; Vistro	0%	Stop bar detectors are available	

[a]The "required minimum penetration rate" refers to the minimum penetration rate where the proposed method can outperform the performance of its benchmarks in the experimental environment. For studies that do not provide with such information, we use "Not Available" instead.

Table 2
Existing research of RL based TSC with low penetration rates.

Reference	Information source	Environment	Agent	Algorithm	Benchmarks	State	Action	Reward	Gap
Aziz et al. (2019)	CVs	A real-world 4-intersection corridor; A real-world network of 20 intersections	TSC of one intersection	Q-learning	The proposed algorithm with different rewards	Estimated queue lengths	Keep current phase or change to the next phase	Delay; Energy consumption; Or energy consumption with penalty for stops	(1) An RL based TSC system using CV data and specifically designed for scenarios with low penetration rates is missing; (2) It remains unclear what CV information could improve robustness against partial observation from CVs; (3) The RL structure suitable for partially observed system remains to be explored.
Wu et al. (2020)	CVs, camera	A 2-3 synthetic[a] network	TSC of one intersection	MARDDPG	Fixed-time; DDPG; SOTL; IDQN	Vehicle location; Vehicle velocity; Queue length; Current traffic light phase; Number of pedestrians	Keep current phase or change to the next phase	Weighted sum of delay; Queue length; Throughput; Blinking condition of traffic; Waiting time of vehicles and pedestrians	
Zhang et al. (2020)	CVs	A synthetic isolated intersection	TSC of one intersection	DQN	Agent trained and tested under the same scenario; Fixed-time TSC	The distance to the nearest detected vehicle; Number of detected vehicles; Amber phase indicator; Current traffic light Current phase elapsed time; Current time of a day; Current phase	Keep current phase or change to the next phase	Negative value of speed loss	
CVLight (this paper)	CVs	A 2-2 synthetic network; A 5-5 synthetic network; A 2-2 real-world network	TSC of one intersection	Asym-A2C[b]	Max-Pressure PressLight DQN Webster's method Actuated TSC	Number of vehicles in each incoming and outgoing lane; Average delay per vehicle of each incoming lane; Current traffic signal phase; Current phase elapsed time	Choose the next phase	Negative value of pressure on the intersection	

[a]The "2-3" represents the 2-by-3 road network structure.
[b]We will introduce the Asym-A2C in Section 2.4.

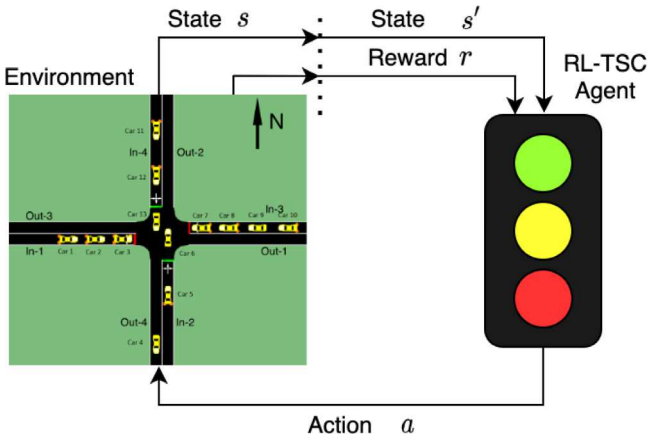


Fig. 1. An illustration of the interaction between RL-TSC agent and traffic environment.

Specifically, the main contributions of this paper are listed below:

1. We model the TSC system that leverages CVs information using a partial observable reinforcement learning scheme, denoted as CVLight. We design states and rewards for CVLight to foster inter-agent coordination and consider delays and phase durations.
2. We develop a novel training algorithm, Asymmetric Advantage Actor Critic (Asym-A2C), which utilizes asymmetric information: the actor is trained with the partial observation of CVs while the critic is trained with the full observation of both CVs and non-CVs.
3. We demonstrate the advantage of CVLight over benchmarks using a comprehensive list of numerical experiments. We also demonstrate that, by utilizing pre-trained models of small road networks, CVLight can be applied to larger road networks at lower computation costs.

The remainder of this paper is structured as follows. Section 2 presents the basics of reinforcement learning in the context of TSC with CVs and outlines the CVLight model. The proposed model is then examined in Section 3 to validate the design, including performance comparison, sensitivity analysis, and scalability in road networks sizes. In Section 4, CVLight is compared to multiple benchmarks on real-world intersections. Finally, we conclude the paper and present future research directions in Section 5.

2. Reinforcement Learning based Traffic Signal Control (RL-TSC)

In this section, we will introduce concepts and terminologies in reinforcement learning based traffic signal control (RL-TSC), taking one state-of-the-art RL-TSC algorithm known as PressLight (Wei et al., 2019) as an example. Based on this, we discuss the limitation of Presslight under scenarios without a full penetration rate and propose a new model, *CVLight*.

Prior to delving into the introduction of RL and our proposed model, we first define major notations that will be used in the subsequent sections.

2.1. Preliminaries

For an isolated intersection, we regard the control unit of traffic signals as an *agent*, and all other things as *environment*. The TSC problem can be formulated as a Markov decision process (MDP), which is specified by a tuple of (S, A, R, Pr, γ). The state space S contains all necessary information describing the environment, such as the queue length of each lane. A is a set of actions that an agent can perform to interact with the environment, e.g. to determine the length of the next traffic signal phase. Given a certain state $s \in S$, the agent chooses one action $a \in A$ following its policy $\pi(a|s)$, which maps from the state to actions in the action space A.

The state transition function Pr is a perfect model of the environment's mechanism. As shown in Fig. 1, once the newly chosen action is executed, the agent can observe a new state s' based on Pr and receive a reward $r \in R$. The goal of RL is to find the optimal policy π^* that can maximize the return, which is a function of a sequence of rewards weighted by a discounted factor γ. To estimate the expected return under a certain circumstance, two kinds of value functions are proposed: a state value function $V^\pi(s)$ estimates how favorable it is to be in the state s under policy π; and a state–action value function $Q^\pi(s, a)$ estimates how favorable it is to take action a from state s under policy π. For a complicated real-world traffic environment, the state transition function Pr is unknown. In this case, the MDP problem can be solved by Temporal Difference (TD) techniques, such as Q-learning and Policy Gradients. More details about RL can be found in Sutton et al. (1998).

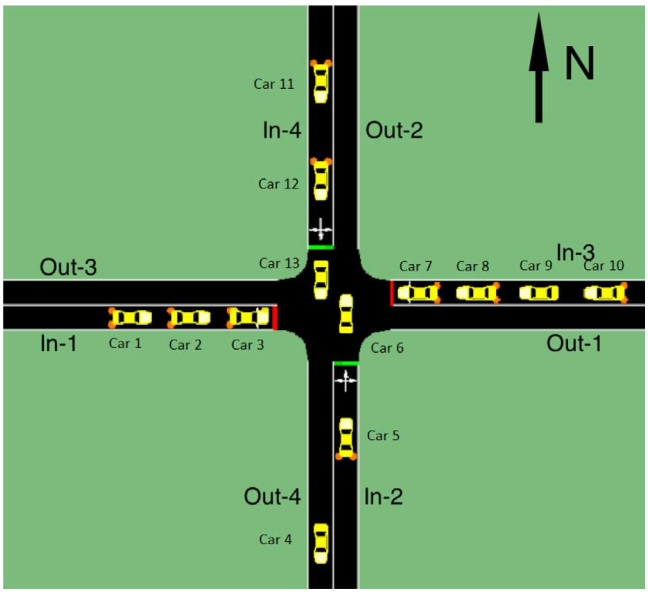


Fig. 2. An example of the state for PressLight.

To better illustrate how an RL-TSC problem is formulated, we use PressLight (Wei et al., 2019) as an example. Theoretically grounded by a popular TSC algorithm, Maxpressure (Varaiya, 2013), PressLight is proven to outperform Maxpressure and other recent RL-TSC algorithms, such as GRL (Van der Pol and Oliehoek, 2016), under multi-intersections control scenarios.

The state of PressLight is defined as a vector of the current traffic signal phase and the number of vehicles in each incoming and outgoing lane. For the intersection shown in Fig. 2, the state vector can be represented as:

$$s = [\overbrace{(1,0)}^{\text{phase}}, \underbrace{(3,1,4,2)}_{\substack{\text{number of vehicles}\\ \text{in incoming lanes}}}, \overbrace{(0,0,0,1)}^{\substack{\text{number of vehicles}\\ \text{in outgoing lanes}}}],$$

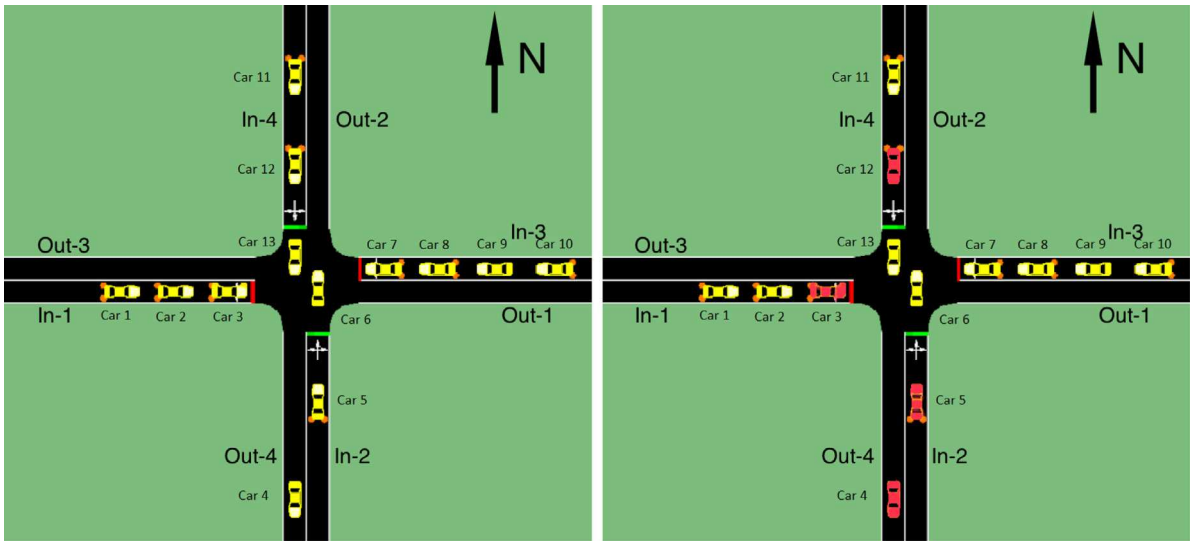
where the first entry of s, a one-hot vector, represents the current green phase for the North–South direction (denoted by NS-allow phase); the vector $(3,1,4,2)$ refers to the numbers of vehicles in incoming lanes from incoming lane 1 (denoted by In-1) to In-4, respectively; similarly, each entry of the vector $(0,0,0,1)$ represents the number of vehicles in the corresponding outgoing lane, from Out-1 to Out-4 (Car 6 and 13 are not counted as they are not in any incoming or ongoing lane).

Observing the current state s, the PressLight agent chooses its action a. The action is an index of one signal phase that will be executed as the next phase. If the index is the same as the index of the current phase, the current phase will continue for a pre-defined length of time, e.g. 7 s; if not, TSC will first execute a yellow phase and an all-red clearance phase consecutively for a pre-defined length of time and then execute the chosen traffic signal phase for the minimum green time.

After the execution of action a, the agent can observe a new state s' and receive a reward r, which indicates how good its chosen action is. For PressLight, the reward is calculated based on the current *pressure*, which is defined as the sum of the difference between the numbers of vehicles in incoming lanes and outgoing lanes of all phases and is presented in Eq. (2.6). For simplicity, we use the traffic state s presented in Fig. 2(a) as an example for pressure calculation. Assuming no turning traffic flows, the pressure of the NS-allow phase is the sum of the difference between the number of vehicles in In-2 and Out-2 and the difference between the number of vehicles in In-4 and Out-4. Thus, the pressure of NS-allow phase is 3. Similarly, the pressure of EW-allow phase is 7 and the total pressure of this intersection is 10 (here we do not consider the lane capacity in Eq. (2.6)).

2.2. CVLight for a single intersection

The previous example in Section 2.1 is a simple scenario where the RL agent can obtain complete information about the environment. However, it might not work in the CV setting. When only partial traffic information is known, the problem becomes a Partially Observable Markov Decision Process (POMDP). In POMDP, agents cannot obtain complete information about the environment's state; instead, they can only obtain partial information, denoted by observation $o \in O$, where O is the observation space. It is not appropriate to directly apply PressLight to scenarios with CV penetration rates below 100%. For example, in Fig. 3(a), if only four vehicles are CVs and the number of vehicles in observation can only be measured based on CVs, then the observation o can be significantly different from the state s. Under such a circumstance, the PressLight agent is not able to detect the congestion on In-1 and In-3 and may learn wrong policies due to the significant difference between states and observations.



(a) State (all cars are observable) (b) Observation (only red cars are CVs)

Fig. 3. An example of state and observation for CVLight (Note that In-1, for example, represents the incoming lane 1 and Out-1 represents the outgoing lane 1).

To cope with the issues mentioned above, we propose *CVLight* and introduce its state, action, and reward design in the single-intersection TSC problem.

(1) **State**. The state s is defined as a vector of current phase p, elapsed time of current phase (named as phase duration d), number of vehicles in each incoming lane $n(l)$ ($l \in L_{in}$, L_{in} is a set of incoming lanes), number of vehicles in each ongoing lane $n(m)$ ($m \in L_{out}$, L_{out} is a set of outgoing lanes), and delay ratio $x(l)$ ($l \in L_{in}$), which is the ratio of the average delay per vehicle of each incoming lane and the total minimum travel time of all vehicles in the incoming lanes. In summary, the state s is written as below:

$$s = [\overbrace{p}^{phase}, \underbrace{d}_{\text{phase duration}}, \overbrace{n(l)}^{\text{number of vehicles in incoming lanes}}, \underbrace{n(m)}_{\text{number of vehicles in outgoing lanes}}, \overbrace{x(l)}^{\text{average delay per vehicle in incoming lanes}}], \ for\ l \in L_{in}, m \in L_{out}. \tag{2.1}$$

Below we will describe how the value of each component is obtained.

- The phase p is a one-hot vector that indicates the current traffic phase at the intersection.
- Due to the limited available information from CVs, if the RL agent cannot observe CVs in one direction, it is possible that the RL agent maintains the green light in another direction for a very long time, which is unrealistic and unfair. Considering this, we include phase duration d into the state, which enables the agent to build a connection between the phase duration and the observed delay of vehicles.
- Similar to PressLight (Wei et al., 2019) and Maxpressure (Varaiya, 2013), we include the number of vehicles in each incoming and outgoing lane as part of the state. Specifically, $n(l)$ is the vehicle count in a certain lane within a given detected radius. In this paper, the detected radius is 200 m. If the lane length is shorter than this default radius, the radius will be adjusted to the actual lane length. We divide the road into three segments for each lane so as to provide spatial distribution information of vehicles to the agent (Yang and Menendez, 2018; Wei et al., 2019).
- Travel delay of a vehicle is defined as the difference between the current travel time and the expected minimum travel time of lane l, denoted by $t_{min}(l)$. The minimum travel time of all vehicles in that lane and is defined as

$$t_{min}(l) = \frac{length(l)}{v_{max}(l)}, \tag{2.2}$$

where

  - $length(l)$ is the length of lane l;
  - $v_{max}(l)$ is the speed limit of lane l.

Therefore, the scaled average delay per vehicle in incoming lane l, $x(l)$, is defined as

$$x(l) = \frac{\sum_{v \in V(l,t)} \left(t - t_0(v,l) - t_{min}(l)\right)}{n(l)\ t_{min}(l)}, l \in L_{in}, \tag{2.3}$$

where

- $V(l,t)$ is the set of all vehicles in lane l at time step t;
- t is the current time step;
- $t_0(v,l)$ is the time step when vehicle v enters lane l.

Since the delay is cumulative over time, it would contain more historical information than the number of vehicles in a lane, even with limited CV data available. Thus, we add the average delay per vehicle into our state as well to better utilize the CV information, especially for low penetration rate scenarios.

(2) **Observation**. We assume that CVs will send their real-time locations and IDs using V2I technologies to both the upstream and downstream intersections. Partial information that is observed by CVs is denoted below (note that we include the traffic signal information, p and d, into o_{CV} as well):

$$o_{CV} = [\overbrace{p}^{phase}, \underbrace{d}_{\text{phase duration}}, \overbrace{n_{CV}(l)}^{\text{number of CVs in incoming lanes}}, \underbrace{n_{CV}(m)}_{\text{number of CVs in outgoing lanes}}, \overbrace{x_{CV}(l)}^{\text{average delay per CV in incoming lanes}}], \ for\ l \in L_{in}, m \in L_{out}. \tag{2.4}$$

Partial information of non-CVs (denoted by $o_{\overline{CV}}$), which is not observed by CVs, is denoted as:

$$o_{\overline{CV}} = [\overbrace{n_{\overline{CV}}(l)}^{\text{number of non-CVs in incoming lanes}}, \underbrace{n_{\overline{CV}}(m)}_{\text{number of non-CVs in outgoing lanes}}, \overbrace{x_{\overline{CV}}(l)}^{\text{average delay per non-CV in incoming lanes}}], \ for\ l \in L_{in}, m \in L_{out}. \tag{2.5}$$

(2) **Action**. Traffic signal phases are acyclic. Once the RL agent observes o_{CV}, it can select one phase p as its action a from its phase set (action set) A. If p is the same as the current phase, the current phase will continue for a certain period of time (in this paper, we use the minimum green time, which is 7 s); or if p is different from the current phase, the intersection will first consecutively experience a yellow phase (1 s) and an all-red clearance phase (2 s), and then execute the chosen phase p for a minimum green time. To avoid extreme behavior such as keeping one phase for a long time, we enforce the maximum green time (40 s by default) for phase duration. Once the current phase duration exceeds the maximum green time, the agent will be enforced to choose the phase with the highest probability apart from the current phase as the next phase.

(3) **Reward**. The reward r adopts the reward from PressLight (Wei et al., 2019), which is the negative value of the pressure:

$$r = -P = -\left| \sum_{(l,m)\in L_{phase}} \left(\frac{n(l)}{n_{max}(l)} - \frac{n(m)}{n_{max}(m)} \right) \right|, \tag{2.6}$$

where

- P is the pressure;
- L_{phase} is a set of combinations of incoming lane $l \in L_{in}$ and outgoing lane $m \in L_{out}$ of each traffic movement controlled by traffic signal phases;
- $n(l)$ is the number of vehicles in lane l;
- $n_{max}(l)$ is the capacity of lane l, i.e. the maximum number of vehicles the lane can store.

2.3. CVLight for multiple intersections

We will introduce the agent design of *CVLight* for scenarios of multiple intersections.

(1) **State**.

$$s_i = [\overbrace{p_i}^{phase}, \underbrace{d_i}_{\text{phase duration}}, \overbrace{n_i(l)}^{\text{number of vehicles in incoming lanes}}, \underbrace{n_i(m)}_{\text{number of vehicles in outgoing lanes}}, \overbrace{x_i(l)}^{\text{average delay per vehicle in incoming lanes}}], \ for\ l \in L_{in}(i), m \in L_{out}(i), \tag{2.7}$$

where the subscript is the index of each intersection. $L_{in}(i)$ and $L_{out}(i)$ are sets of incoming and outgoing lanes of the intersection i, respectively.

(2) **Observation.**

$$o_{i,CV} = [\overbrace{p_i}^{phase}, \underbrace{d_i}_{\text{phase duration}}, \overbrace{n_{i,CV}(l)}^{\text{number of CVs in incoming lanes}}, \underbrace{n_{i,CV}(m)}_{\text{number of CVs in outgoing lanes}}, \overbrace{x_{i,CV}(l)}^{\text{average delay per CV in incoming lanes}}], \ for\ l \in L_{in}(i), m \in L_{out}(i). \tag{2.8}$$

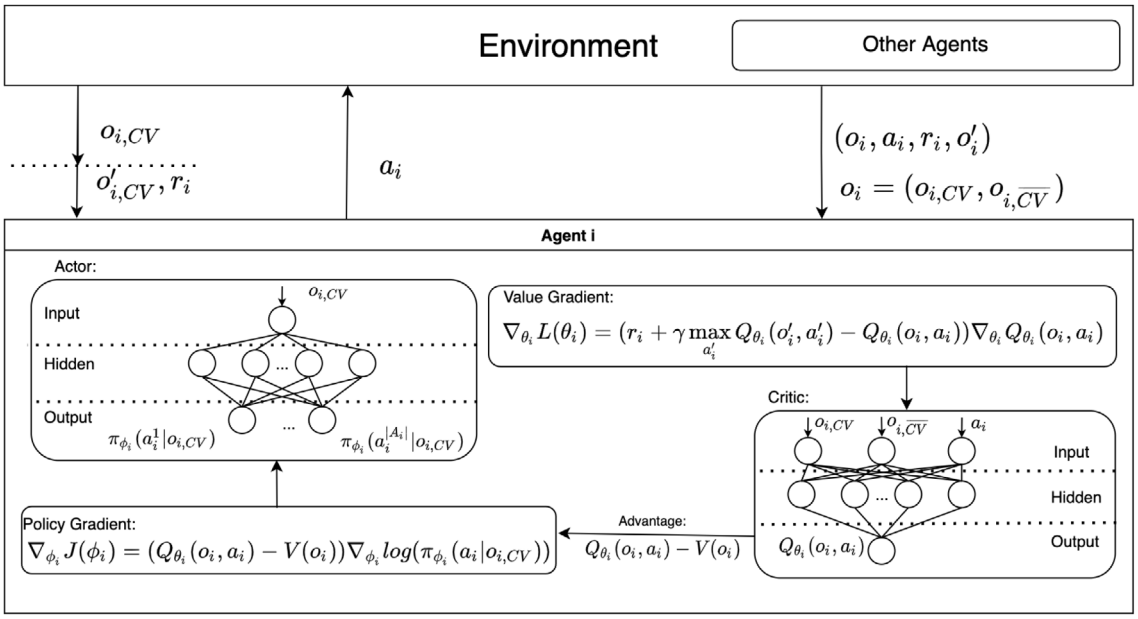


Fig. 4. An illustration of the Asymmetric Advantage Actor–Critic (Asym-A2C) algorithm.

$$o_{i,\overline{CV}} = [\ \overbrace{n_{i,\overline{CV}}(l)}^{\text{number of non-CV in incoming lanes}}\ ,\ \underbrace{n_{i,\overline{CV}}(m)}_{\text{number of non-CVs in outgoing lanes}}\ ,\ \overbrace{x_{i,\overline{CV}}(l)}^{\text{average delay per non-CV in incoming lanes}}\ ],\ for\ l \in L_{in}(i), m \in L_{out}(i). \tag{2.9}$$

(2) **Action**. Intersection i chooses action a_i independently, once it observes the $o_{i,CV}$. Note that agents take actions asynchronously, and thus the time steps when agents make new actions can be different from each other.

(3) **Reward**. The reward of each intersection is the same as in the single-intersection setting as in Eq. (2.6), and we use the subscript to indicate the reward of the intersection i:

$$r_i = -P_i = -\left| \sum_{(l,m)\in L_{phase}(i)} \left(\frac{n_i(l)}{n_{max}(l)} - \frac{n_i(m)}{n_{max}(m)} \right) \right|, \tag{2.10}$$

where

- P_i is the pressure of intersection i;
- $L_{phase}(i)$ is a set of combinations of incoming lane $l \in L_{in}(i)$ and outgoing lane $m \in L_{out}(i)$ of each traffic movement controlled by traffic signal phases at intersection i;
- $n_i(l)$ is the number of vehicles in lane l;
- $n_{max}(l)$ is the capacity of lane l, i.e. the maximum number of vehicles the lane can store.

Using pressure as the reward is proved to be able to guide agents to maximize the throughput of the system and thus minimize the travel time of all vehicles (Wei et al., 2019). Cooperation among agents is achieved by incorporating traffic information not only from incoming lanes but also from outgoing lanes into the reward calculation, which naturally connects the upstream and downstream intersections.

2.4. Training algorithm: Asymmetric Advantage Actor–Critic method

In this subsection, we will propose a new training algorithm for CVLight, the *Asymmetric Advantage Actor–Critic* (Asym-A2C) method, which can be viewed as an asymmetric variant of the Advantage Actor–Critic method (Mnih et al., 2016).

The novelty of Asym-A2C is to leverage its actor–critic architecture to feed asymmetric inputs into the critic and the actor: the critic is fed with full state information while the actor's access is restricted to partially observed information from CVs.

Fig. 4 details our implementation of Asym-A2C in a multi-intersection scenario where all agents are independent of each other and own the exact same structure as agent i (Shou and Di, 2020; Shou et al., 2022). For the Asym-A2C algorithm, the critic, parametrized by θ, estimates the value function (state value or the state–action value), and the actor subsequently updates its parameters ϕ in the

direction suggested by the critic. Deep neural networks are used to estimate the value function for the critic as well as the policy model for the actor. For agent i, it observes the full observation o_i, which is composed of observation from CVs (i.e. $o_{i,CV}$) and that from non-CVs (i.e. $o_{i,\overline{CV}}$). For the actor, we assume it can only receive information from CVs, i.e. the observation for the actor is $o_{i,CV}$. Based on the policy network ϕ_i, that the actor then chooses one action a_i from its action space A_i (we denote the kth action in the action space by a_i^k where $k \in \{1, 2, \dots, |A_i|\}$ and $|A_i|$ is the number of all possible actions). After the execution of a_i, the agent can receive a new observation o_i' as well as a reward r_i. An experience tuple (o_i, a_i, r_i, o_i') is then stored into the experience replay buffer of agent i. In the same way, other agents in the environment store experience tuples into their experience replay buffers independently.

We assume the training process is off-line and the critic can receive information from both CVs and non-CVs, i.e. o_i includes $o_{i,CV}$ and $o_{i,\overline{CV}}$. During training, experience tuples are sampled from the agent's experience replay buffer and used to update the critic network using the following gradient:

$$\nabla_{\theta_i} L(\theta_i) = (r_i + \gamma \max_{a_i'} Q_{\theta_i}(o_i', a_i')) \nabla_{\theta_i} Q_{\theta_i}(o_i, a_i) \tag{2.11}$$

where

- γ is the discounted factor;
- a_i' is the next action that maximizes the value function given the next observation o_i'.

Each update on the parameter of the value network θ_i will allow the agent i to better estimate the value function given the observation and action.

The policy network is then updated using the gradient computed as follows:

$$\nabla_{\phi_i} J(\phi_i) = \mathbb{E}_{\pi_{\phi_i}} [\nabla_{\phi_i} \log \pi_{\phi_i}(a_i | o_{i,CV}) A(o_i, a_i)], \tag{2.12}$$

where $A(o_i, a_i) = Q_\theta(o_i, a_i) - V(o_i)$ is called the *advantage* of action a_i in the observed state o_i. Intuitively, it can be treated as a measure of how much better action a_i is compared to the average. $V(o_i)$ is the baseline, which reduces the gradient size and leads to a much lower variance estimate of the policy gradient (Mnih et al., 2016). Each update on the policy network ϕ_i will increase the probability of choosing a suitable action so as to achieve a $Q_\theta(o_i, a_i)$ higher than the average state value function $V(o_i)$ for the observation o_i.

Now, we summarize key ideas of Asym-A2C as follows:

1. In the training stage, each critic can evaluate the actor's behavior using information from both CVs and non-CVs at each intersection, while each actor can only observe CV data. Specifically, the input of the critic is composed of traffic observations of CVs and non-CVs. But for the actor, it still can only get information from CVs. To some degree, the critic helps the actor to build a connection between observed CV data and the traffic states of all vehicles;
2. In the execution (or test) stage, actors select their best actions following the learned policies solely using CV data.

As a comparison, we also propose the *Symmetric A2C* (denoted by Sym-A2C) where both the actor and the critic receive the same input observation from CV only. We denote the CVLight that utilizes the Asym-A2C and Sym-A2C as *CVLight (Asym)* and *CVLight (Sym)*, respectively. To make a fair comparison, we allow both the CVLight (Asym) agent and the CVLight (Sym) agent to receive rewards based on the traffic states of CVs and non-CVs.

As to the settings of our neural networks, for each CVLight agent, the policy neural network consists of two hidden layers. Each layer consists of $2|o_{CV}|$ fully connected neurons and exponential linear unit (ELU) activation functions, where $|o_{CV}|$ is the length of the partial observation vector o_{CV}. The value neural network is composed of two hidden layers. Each layer consists of $2|o|$ fully connected neurons and ELU activation functions. An Adam optimizer (Kingma and Ba, 2014) with a learning rate 1e−4 is adopted for both value and policy networks. The experience replay buffer size, batch size, and discount rate γ are 10,000, 128, and 0.99, respectively.

3. Numerical experiments

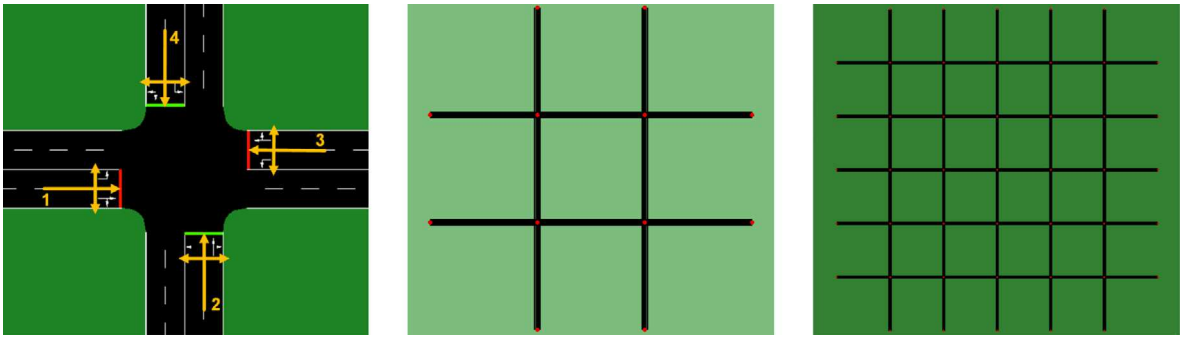
In this section, we conduct experiments under synthetic road networks and compare the proposed CVLight model with other benchmark models. Experiment settings are introduced in Section 3.1. In Section 3.2, we investigate the performance of CVLight and benchmarks under different traffic demand levels and penetration rates. Based on this, we visualize and interpret the learned policies of CVLight in Section 3.3. Furthermore, the sensitivity analysis on the maximum green time (denoted by gmax) and the minimum green time (denoted by gmin) is conducted in Section 3.4. We then discuss the scalability in road network sizes in Section 3.5.

Hyperparameters in different experiments are summarized in Table 3. The first and second columns list different experiments and corresponding road networks. The remaining columns (3 to 6 for train and 7 to 10 for test) are values of different hyperparameters. All unmentioned hyperparameters are kept as default, such as the 10% left turn and 10% right turn proportion. We use bold values to specify the difference of hyperparameters in train and test procedures. In each subsection, we will come back to this table and explain the experiment setting in detail.

Table 3
Settings of numerical experiments.

Experiments	Road networks	Train				Test			
		Demand[a]	Gmin (s)	Gmax (s)	pen[b] (%)	Demand	Gmin (s)	Gmax (s)	pen (%)
Generalizability in traffic demands	2-2	**Dynamic**	7	40	{10,20,30, 50,70,100}	**Dynamic, 600,800**	7	40	{10,20,30, 50,70,100}
Generalizability in penetration rates	2-2	600	7	40	**10**	600	7	40	**{10, 20, 30, 50, 100}**
SA[c] on gmin	2-2	600	{5,7,10}	40	{10,30}	600	{5,7,10}	40	{10,30}
SA on gmax	2-2	600	7	{40,60,120}	{10,30}	600	7	{40,60,120}	{10,30}

[a]Numbers here represent fixed demands in the unit of vehicles/hour/lane. "Dynamic" means the demand is changing according to a fixed pattern, which is detailed in Table 4 and Fig. 6.
[b]"pen" here represents the CV penetration rate.
[c]"SA" here represents sensitivity analysis.



(a) The layout of an intersection (b) 2-2 road network layout (c) 5-5 road network layout

Fig. 5. Illustration of the road network.

3.1. Experiment settings

In this subsection, we introduce the setting of our numerical experiments, including our environment set-up, the measurement of performance, and benchmark algorithms.

3.1.1. Environment set-up

Road Network

We first introduce a synthetic isolated intersection, based on which we can introduce larger road networks. Fig. 5(a) illustrates the structure of a 4-way intersection with two incoming and two outgoing lanes on each way. For each way, each incoming lane consists of one go-straight and right-turn lane and one left-turn lane. The length of each lane is 200 m and CVs can be detected by the RL-TSC agent when they are within 200 m of the intersection.

A 2-phase signal timing plan is used: one phase is the green phase for traffic movement 1 and 3 (represented by arrows in Fig. 5(a)); the other is the green phase for traffic movement 2 and 4. Note that traffic movement here is defined as the traffic that is allowed to pass the intersection under a certain phase and travels in the same direction.

Two road networks, 2-2 and 5-5, are configured. Each intersection in the road networks follows the same 2-phase signal timing plan as the isolated intersection shown in Fig. 5(a). Fig. 5(a) illustrates the 2-2 network of 4 intersections and Fig. 5(c) illustrates the 5-5 network of 25 intersections. We set the East–West (EW) roads as the arterials and the North–South (NS) roads as the side-streets. The length of each lane in the road network is 200 m.

Traffic Demand

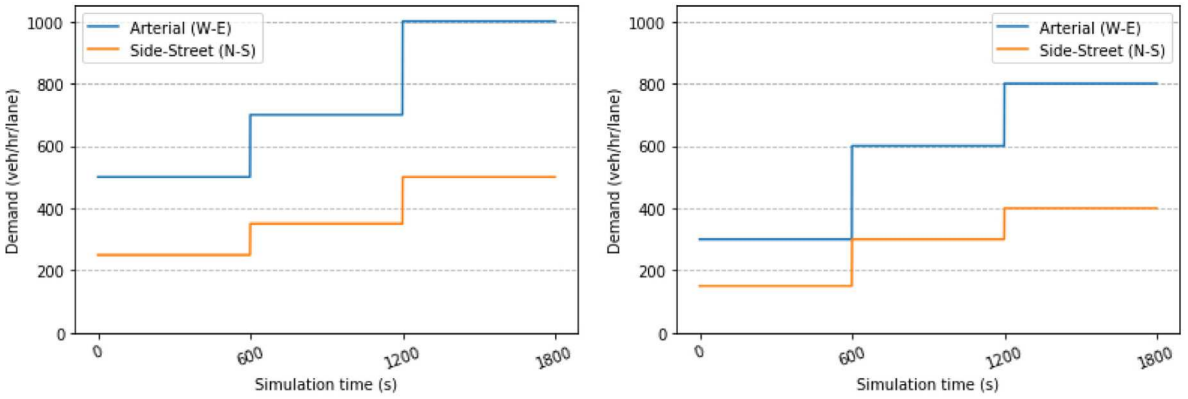
All our experiments use synthetic traffic demand data, as shown in Table 4 and Fig. 6. For Table 4, the first column lists the traffic demand patterns. The second and third columns detail the traffic demand on the arterial and side-street. Fig. 6 presents the dynamic traffic demand patterns, which correspond to the first and second rows in Table 4. The y-axis represents demands and x-axis represents the simulation times. Blue line and orange line mean the demands on the arterials and side-streets, respectively.

For each stage in the dynamic traffic demand, the simulation length is 600 s. For a fixed demand, the simulation length is 1800 s. Vehicles' inter-arrival times follow a binomial distribution by default, i.e. the vehicle arrival process follows a Poisson process. For experiments in this section, both left-turn and right-turn proportions are set to be 10%. Given a penetration rate, we randomly

Table 4
Traffic demands.

Demand patterns	Arrival rate (vehicles/h/lane)	
	Arterial (EW)	Side-street (NS)
Dynamic (train)	500-700-1000[a]	250-350-500
Dynamic (test)	300-600-800	150-300-400
Fixed	600	300
	800	400

[a]The series of numbers represents the demands in different stages in the dynamic traffic demand, which is illustrated in Fig. 6. For example, 500-700-1000 means the traffic demand starts from 500 veh/h/lane, increases to 700 veh/h/lane, and ends with 1000 veh/h/lane.



(a) Dynamic traffic demands (train) (b) Dynamic traffic demands (test)

Fig. 6. Dynamic traffic demand patterns. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

assign vehicles to be CVs based on an equal probability of selection. Vehicles that are not chosen to be CVs are the non-CVs in the simulation.

SUMO Simulation

We conduct experiments in SUMO, an open-source, microscopic, and time-discrete and space-continuous traffic simulation package designed to handle large networks (Behrisch et al., 2011; Krajzewicz et al., 2012; Lopez et al., 2018). The simulation of our experiments is round-based, and the duration of each round is 1800 s by default in this paper.

3.1.2. Performance measurement

To measure the performance of a given model in one round of simulation in the execution stage, we use the average travel delay per vehicle per intersection in a road network as the metric, denoted by M_{delay}, formulated as:

$$M_{delay} = \frac{1}{|I|}\sum_{i\in I}\bar{D}_i, \tag{3.1}$$

$$\bar{D}_i = \frac{1}{|T|}\sum_{t=0}^{|T|}\left(\frac{D_{i,t}}{\sum_{l\in L_{in}(i)} n_i(l,t)}\right), \tag{3.2}$$

$$D_{i,t} = \sum_{l\in L_{in}(i)}\sum_{v\in V(l,t)}(t - t_0(v,l) - t_{min}(l)), \tag{3.3}$$

where

- $|I|$ is the number of intersections on the road network and I is a set of all intersections in the road network;
- $\bar{D}_i$ is the average travel delay per vehicle at the intersection i;
- $|T|$ is the running time of one round in the simulation;
- $L_{in}(i)$ is the set of incoming lanes at intersection i;
- $n_i(l,t)$ is the number of vehicles in lane l at time t;
- $D_{i,t}$ is cumulative travel delay of all vehicles in all incoming lanes at intersection i at time t;
- $V(l,t)$ is the set of all vehicles in lane l at time t;
- t is the time step;

– $t_0(v, l)$ is the time step when vehicle v arrives at lane l;
– $t_{min}(l)$ is the expected minimum travel time of lane l.

The final measurement of performance is the mean and standard deviation of M_{delay} of all test rounds. We conduct 96 rounds of tests by default for all experiments.

3.1.3. Benchmarks

To demonstrate the performance of CVLight, we compare it with various popular benchmark TSC algorithms, including both RL based and non-learning based TSC algorithms.

- **Max-pressure.** Max-pressure (Varaiya, 2013) algorithm is a state-of-the-art network-level traffic signal control method, which greedily chooses the phase with the maximum pressure.
- **Adaptive Webster.** Webster's method (Webster, 1958) is one of the classic TSC methods and is widely applied in the real world. To cope with dynamic traffic demands, an adaptive version is developed (Genders and Razavi, 2019), where the algorithm would periodically update its parameters using the most recent traffic statistics.
- **Deep Q-Network (DQN) based TSC.** DQN (Mnih et al., 2015) is a classic reinforcement learning algorithm, using a deep neural network to estimate the Q-value function. We adopt this algorithm from Genders and Razavi (2019).
- **PressLight.** PressLight (Wei et al., 2019) is another representative state-of-the-art network-level traffic signal control method, based on the Max-pressure.

It is worth noting that these benchmark algorithms are not designed for scenarios without the full knowledge of the traffic. To make a fair comparison, we modify the reward calculation for RL based benchmark algorithms: for both PressLight and DQN-based TSC, the reward is calculated based on all vehicles' information.

3.2. Performance comparison

In this subsection, we compare the performance of CVLight algorithms and that of baselines under different traffic demands and penetration rates. Specifically, we investigate the generalizability of CVLight in demand patterns and penetration rates: agents are trained under one setting and tested under other settings that they have not seen in training, which are detailed in the first and second rows in Table 3.

3.2.1. Generalizability in traffic demands

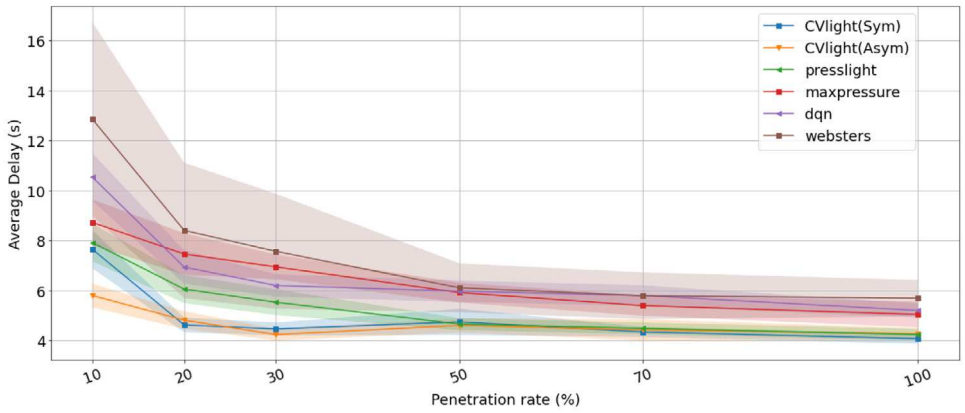
To investigate model generalizability in traffic demands, we train our models under one dynamic traffic demand pattern and test them under another dynamic and two fixed traffic demand patterns, as shown in the first row of Table 3. Fig. 7 compares the performance between CVLight and benchmark algorithms under the 2-2 road network with various penetration rates and traffic demands. In Fig. 7, each sub-figure illustrates the performance of CVLight and benchmark algorithms under a certain test demand. The x-axis is the CV penetration rate and the y-axis represents the average delay per vehicle per intersection. The blue, orange, green, red, purple, and brown lines represent the performance of CVLight (Sym), CVLight (Asym), PressLight, Maxpressure, DQN, and Webster's, respectively. The traffic demand is detailed in Table 4. During tests, all models do not have access to the non-CV information.

From Fig. 7, we can interpret the results from the following two perspectives:

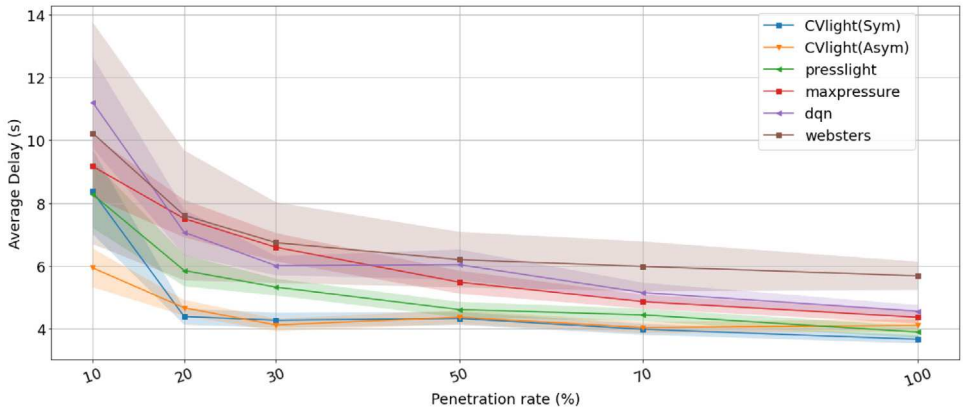
Comparison between CVLight algorithms and other baselines. CVLight agents can achieve the lowest average delay with small standard deviation values under the three test demand levels, compared to benchmark algorithms. As the train and test traffic demands are different, the good performance of CVLight agents indicates that they can generalize well to unseen traffic demand levels. Specifically, under relatively low penetration rates (below 50%), CVLight (Asym) and CVLight (Sym) can achieve the most significant advantage over all other benchmark algorithms across all three test scenarios. One explanation for this is the advantage of adding delay and phase duration information into the state, which contain cumulative information over time and provide a more accurate description of the true traffic state than the number of vehicles does under a low penetration rate.

Comparison between CVLight (Asym) and CVLight (Sym). Across the three test scenarios, under an extremely low penetration rate (10%), CVLight (Asym) shows an advantage over CVLight (Sym). A2C-Asym enables the CVLight agent to better build connections between the observation from CVs only and the state from all vehicles. During training, the critic of CVLight (Asym) can access information from all vehicles and therefore evaluate the action chosen by the actor in a more accurate way.

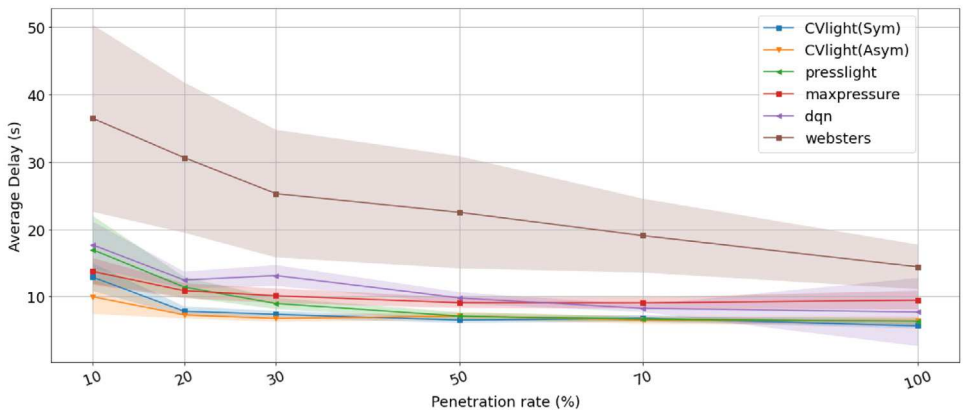
To better describe training process, Fig. 8 shows the training losses of each actor and critic of all CVLight (Asym) agents in the experiment in Fig. 7(a). In Fig. 8, the x-axis is the training iteration index, and the y-axis is the loss value. The blue lines represent the progressions of the training losses of the critics (left column) and actors (right column), each row representing one agent. The dashed black lines and numbers represent the average loss values in the last 1000 iterations. From Fig. 8, all actors and critics converge well.



(a) 2-2 Dynamic



(b) 2-2 600

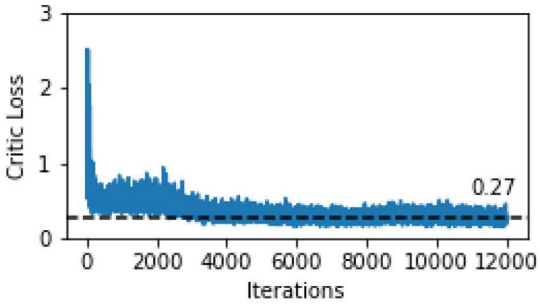


(c) 2-2 800

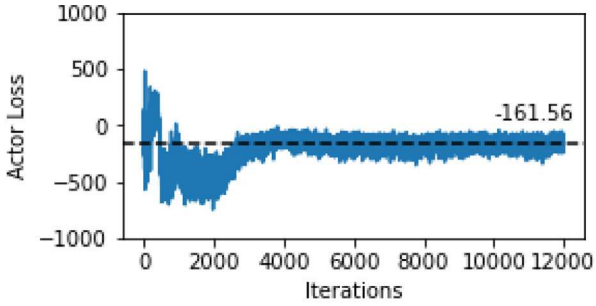
Fig. 7. Performance comparison under a 2-by-2 road network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.2.2. Generalizability in penetration rates

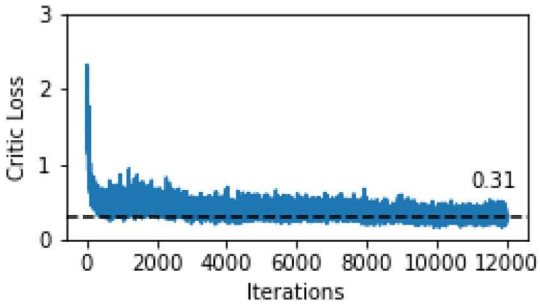
We evaluate the generalizability of our proposed model under unseen penetration rates: we train our models under 10% penetration rate and test them under 10%, 30%, 50%, and 100% penetration rates, as shown in the second row of Table 3. The results are shown in Table 5. The first column lists the names of algorithms and the second to the sixth columns present the average delay achieved by each algorithm under 10%, 20%, 30%, 50%, and 100% penetration rates, respectively. Each entry in the table is the mean value and standard deviation of average delays across all test rounds. We can interpret the results as below:



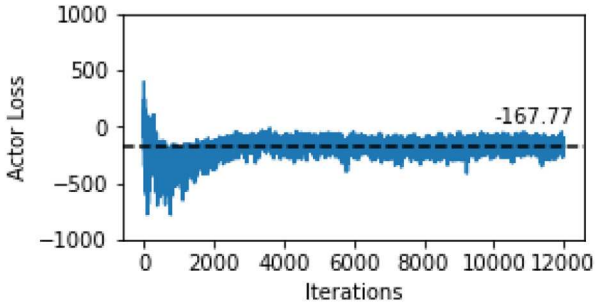
(a) Training losses of critic 1



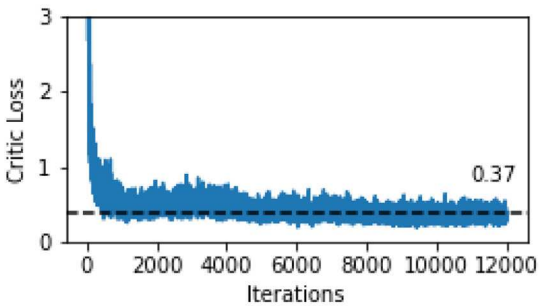
(b) Training losses of actor 1



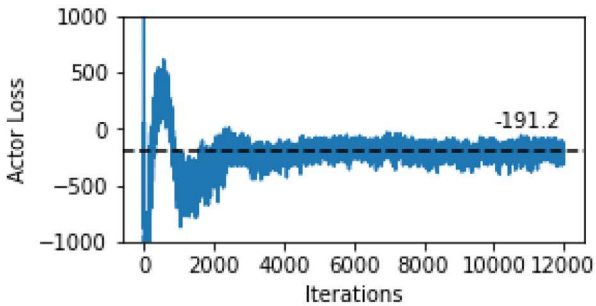
(c) Training losses of critic 2



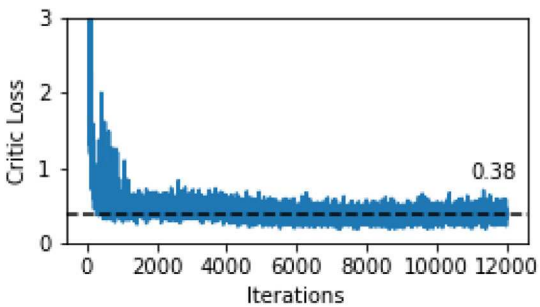
(d) Training losses of actor 2



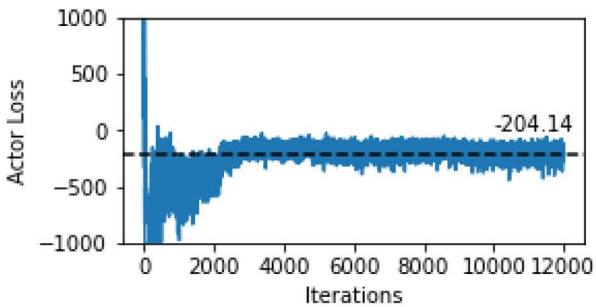
(e) Training losses of critic 3



(f) Training losses of actor 3



(g) Training losses of critic 4



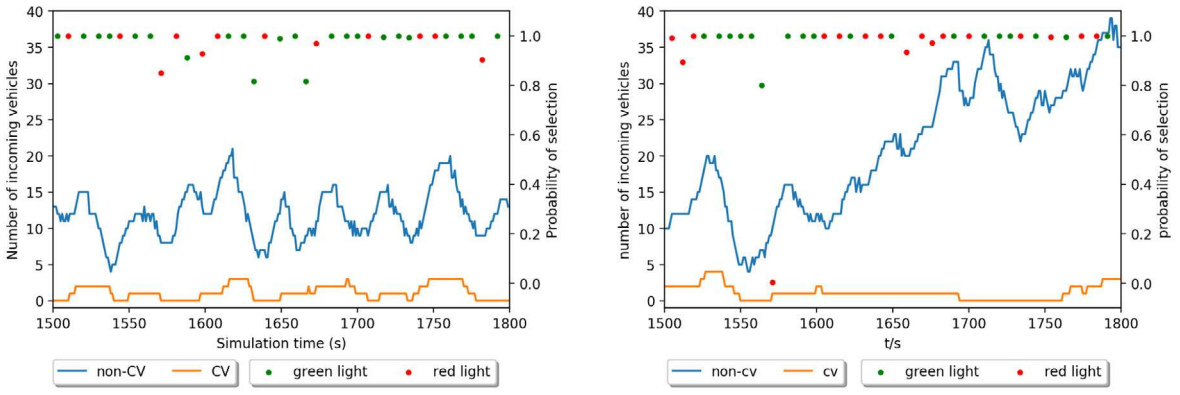
(h) Training losses of actor 4

Fig. 8. Training losses of all CVLight (Asym) agents in the experiment in Fig. 7(a).

CVLight (Asym) and CVLight (Sym) show good generalizability in penetration rates: their performance under an unseen high penetration rate, e.g. 100%, can achieve the same level as their performance under 10%, while PressLight and DQN agents fail to generalize well to such a high penetration rate. The good performance of CVLight can be contributed to the state design, which includes both current traffic information, e.g. the number of vehicles in each lane, and time-cumulative information, e.g. average delay in each lane. CVLights significantly decrease the average delay compared to baselines across all test penetration rates.

Table 5
Performance comparison on generalizability in penetration rate.

Algorithms	CV penetration rate				
	10%	20%	30%	50%	100%
CVLight (Asym)	**4.53** ± 0.36	**4.34** ± 0.30	**4.24** ± 0.30	**4.14** ± 0.25	**4.09** ± 0.27
CVLight (Sym)	4.58 ± 0.39	4.43 ± 0.35	4.33 ± 0.29	4.30 ± 0.29	4.25 ± 0.31
Presslight	4.94 ± 0.33	5.38 ± 0.36	5.53 ± 0.47	6.15 ± 0.65	9.13 ± 1.99
DQN	5.39 ± 0.36	5.62 ± 0.42	5.87 ± 0.99	6.98 ± 4.75	33.37 ± 17.50



(a) CVLight (Asym) learned policy

(b) CVLight (Sym) learned policy

Fig. 9. Policy visualization for CVLight (Asym) and CVLight (Sym) under 2-2 with dynamic traffic demand and 10% penetration rate. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

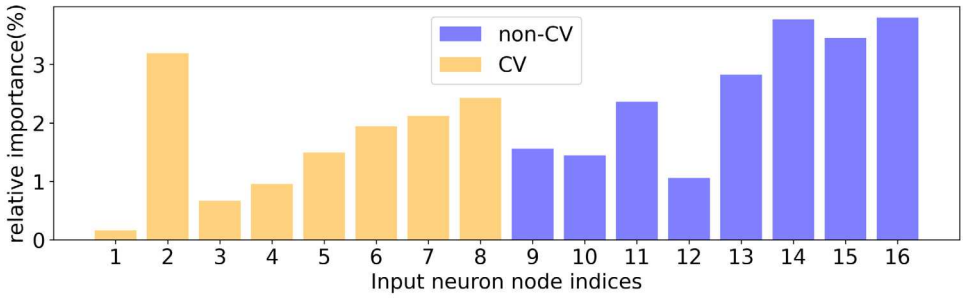
Specifically, CVLight (Asym) achieves the minimal average delay in general across all test penetration rates, which can be contributed to the Asym-A2C.

3.3. Learned policy interpretation

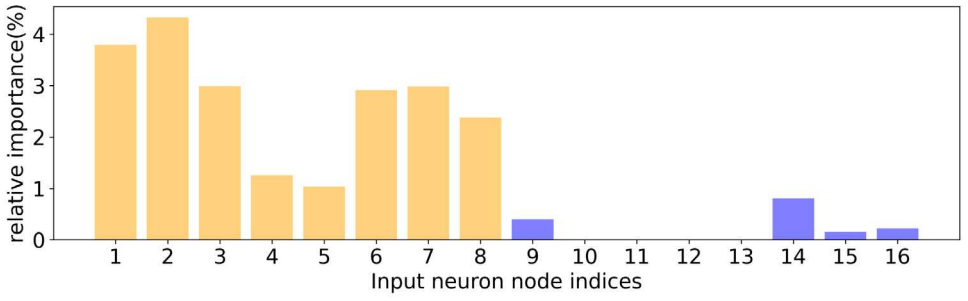
Learned policies of CVLight (Asym) and CVLight (Sym) are illustrated as in Fig. 9. Those policies correspond to the 10% penetration rate case in Fig. 7(a). Fig. 9 presents the visualization of the learned policy of CVLight agents, where the left and right sub-figures are results of CVLight(Asym) and CVLight(Sym), respectively. The x-axis is the simulation time step, the left y-axis is the number of vehicles that are waiting in the incoming lane of the arterial, and the right y-axis is the probability of choosing a green phase or a red phase (i.e. not selecting the green phase) in the arterial direction. Actors of CVLight agents can only observe CVs, i.e. orange lines in the figure, and make decisions based on that information. The green dot or red dot represents the action (i.e. the next phase to be executed) that the agent chose at every decision time and the position of each dot shows the probability of selecting such action based on the learned policy. The red dot at around 1570 s in Fig. 9(b) is caused by the enforcement of the maximum green time (40 s), considering the 6 green dots before the red dot and the 7 s interval between each pair of dots. We can analyze the result as below:

Comparing Fig. 9(a) to (b), CVLight (Asym) agent can better balance the traffic flow in both the arterial and the side-street directions and avoid congestion: it learns to give longer green phases to the arterial direction while switching to the side-street direction for a short time when it observes few CVs in the arterial direction. Specifically, we can compare the actions of CVLight (Asym) and CVLight (Sym) at simulation time 1550 s when both agents do not observe any CVs in the arterial direction: CVLight (Asym) chooses to give green phase to the side-street while CVLight (Sym) continues to give green phase to the arterial until the enforcement of the maximum green time at around 1570 s. Such extreme behavior of CVLight (Sym), i.e. keeping a certain phase for a long time, can explain the performance difference between CVLight (Asym) and CVLight (Sym) under the 10% penetration rate as shown in Fig. 7(a). This demonstrates the effectiveness of Asym-A2C: by incorporating non-CV information into training, CVLight (Asym) agent can learn to make better decisions under a low penetration rate scenario.

To further investigate the effectiveness of Asym-A2C, Fig. 10 illustrates the relative importance (Gevrey et al., 2003) of each delay related neuron in the input layer of the CVLight (Asym) critic neural network. For each input neuron indexed by j, the relative



(a) 10% penetration rate



(b) 100% penetration rate

Fig. 10. Relative importance of delay related neurons in the input layer of the CVLight (Asym) critic neural network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

importance, denoted by $RI(\%)_j$, is presented as below:

$$RI(\%)_j = \frac{\sum_{h=1}^{N_h} Q_{jh}}{\sum_{h=1}^{N_h}\sum_{j=1}^{N_j} Q_{jh}} \times 100, \tag{3.4}$$

$$Q_{jh} = \frac{|W_{jh}|}{\sum_{j=1}^{N_j} |W_{jh}|}, \tag{3.5}$$

where

- j is the input neuron index;
- h is the hidden neuron index;
- N_j is the number of neurons in the input layer;
- N_h is the number of neurons in the next hidden layer;
- $|W_{jh}|$ is the absolute value of the input-hidden layer connection weight between input neuron j and hidden neuron h.

The two sub-figures in Fig. 10 correspond to the experiments in Fig. 7(a) under penetration rates 10% and 100%, respectively. The x-axis represents the indices of delay related neurons in the input layer, where the 1st to 8th neurons relate to the CV delay information (each neuron corresponds to one incoming lane of the intersection) and the 9th to 16th neurons relate to the non-CV delay information. The y-axis represents the relative importance of the inputs. The orange bars represent the relative importance of neurons corresponding to the CV delay and the purple bars represent the relative importance of neurons corresponding to the non-CV delay. To highlight the difference between CV and non-CV delay related neurons, we generate the relative importance using weights with absolute values higher than 0.267, which is the limit of the He uniform weights initialization (He et al., 2015) in our case (the limit is the $\sqrt{6/|o_i|}$, where the $|o_i|$ is the number of neurons in the input layer of the critic neural network of agent i, which is 84 in this experiment). From Fig. 10(a), the non-CV delay related neurons contain higher values of the relative importance than the CV delay related neurons do in general, which shows that non-CV delay information gains more attention from the agent under the 10% penetration rate scenario. Fig. 10(b) shows that the CV related neurons contain higher values in the relative importance than the non-CV related neurons do, which means CV delay information is more representative under the 100% penetration rate. In summary, relative importance of delay related input layer neurons shows that the critic of CVLight (Asym) agent does learn to rely more on the non-CV delay information under a low penetration rate scenario. This also explains the superiority of CVLight (Asym) over CVLight (Sym) under low penetration rate scenarios.

Table 6
Performance comparison under different control intervals (gmin).

Algorithms	Gmin 5		Gmin 7		Gmin 10	
	10%[a]	30%	10%	30%	10%	30%
CVLight (Asym)	**8.27** ± 0.98	**4.48** ± 0.24	**4.53** ± 0.36	**3.82** ± 0.24	**4.35** ± 0.20	**3.92** ± 0.18
CVLight (Sym)	10.8 ± 1.27	5.28 ± 0.38	4.58 ± 0.39	3.90 ± 0.25	4.50 ± 0.25	4.23 ± 0.18
PressLight	10.67 ± 1.80	6.03 ± 0.46	4.94 ± 0.33	4.85 ± 0.35	4.81 ± 0.32	4.75 ± 0.27
DQN	17.22 ± 6.88	11.27 ± 1.50	5.39 ± 0.36	5.04 ± 0.31	9.15 ± 1.21	5.04 ± 0.27
Maxpressure	9.22 ± 0.98	7.11 ± 0.68	9.07 ± 0.95	6.47 ± 0.37	6.98 ± 2.08	5.45 ± 0.48

[a]The percentages here refer to the CV penetration rates.

3.4. Sensitivity analysis

In this subsection, we conduct sensitivity analysis (SA) on our proposed model using numerical experiments. We investigate the robustness of our proposed model under different control intervals (minimum green times) and maximum green times.

All experiments conducted in this section are based on the 2-2 road network with a fixed traffic demand, i.e. 600 veh/h/lane for the arterial and 300 veh/h/lane for the side-street. We only focus on low penetration rates of 10% and 30% where average delays of different models are the most distinct according to Fig. 7.

3.4.1. Control interval

We conduct experiments under 3 different control intervals: 5, 7, and 10 s, which are shown as the third row of Table 3. The results are shown in Table 6. The first column is the names of different models. The remaining columns are organized in two hierarchies. The top level columns are three different values of control intervals. The second level columns are two penetration rates. Each entry is the mean and standard deviation of the average delay after 96 test rounds. For each column, the minimal average delay is in bold. We can interpret the results in three perspectives:

Influence of the control interval on the performance of CVLight. Comparing the performance of CVLight under gmin 5 and gmin 7, a too short control interval, i.e. gmin 5, deteriorates the performance of CVLight under a low penetration rate, as agents switch the phase more frequently than necessary when CVs are not observed. Due to lack of flexibility, CVLight agents also suffer from a too long control interval when they gain more information from CVs, comparing the performance of CVLight under gmin 7 and gmin 10 with the 30% penetration rate.

Comparison between CVLight algorithms and baselines. CVLight (Asym) and CVLight (Sym) significantly decrease the average delay compared to baselines across all control intervals and penetration rates.

Comparison between CVLight (Asym) and CVLight (Sym). CVLight (Asym) achieves the minimal average delay in general. CVLight (Asym) has a notable improvement compared to CVLight (Sym), in which the biggest improvement is 23.4% (5 s Gmin and 10% penetration rate).

3.4.2. Maximum green time

We conduct experiments under 3 different maximum green times: 40, 60, and 120 s, which are shown in the fourth row of Table 3. The results are shown in Table 7. It shares the same structure as Table 6 except for the top level of the columns, which are values of the maximum green time. We can interpret the results from three perspectives:

Influence of the maximum green time on the performance of CVLight. Under the 10% penetration rate scenario, a too long maximum green time can influence the performance of CVLight as agents might continue to give a certain direction green phase for too long when CVs in another direction are not observed. Comparing the performance of CVLight algorithms under the 30% penetration rate across the three gmax values and the its performance under the 10% penetration rate across the three gmax values, we can see that the effect of gmax on the performance will not be that significant when penetration rate gets higher as agents can make better decisions with more information from CVs.

Comparison between CVLight algorithms and baselines. CVLight (Asym) and CVLight (Sym) significantly decrease the average delay compared to baselines across all maximum green times and penetration rates.

Comparison between CVLight (Asym) and CVLight (Sym). Across the three max green times, CVLight (Asym) notably outperforms CVLight (Sym) in terms of the average delay.

3.5. Scalability in road network sizes

In this subsection, we demonstrate the scalability of CVLight in terms of road network sizes by applying pre-trained CVLight for a 5-5 road network.

To further improve the scalability of CVLight, we use a pre-train technique to speed up the convergence in neural networks training. The pre-train technique allows RL agents to load the pre-trained RL models, as long as they share the same neural network

Table 7
Performance comparison under different maximum green times (gmax).

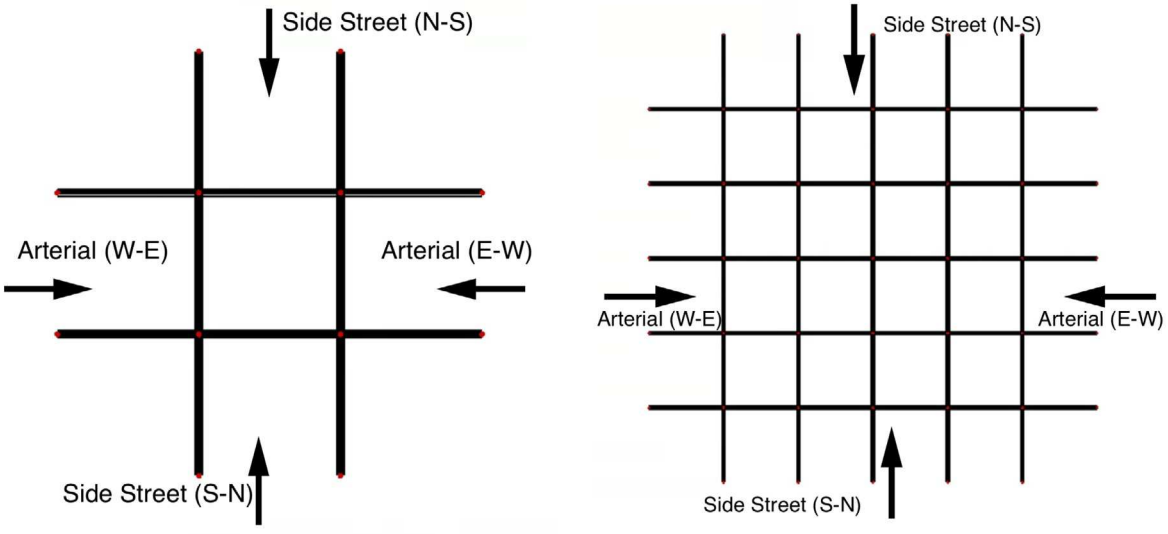
Algorithms	Gmax 40		Gmax 60		Gmax 120	
	10%[a]	30%	10%	30%	10%	30%
CVLight (Asym)	**4.53** ± 0.36	**3.82** ± 0.24	**4.71** ± 0.38	**3.92** ± 0.22	**5.18** ± 0.55	**3.88** ± 0.18
CVLight (Sym)	4.58 ± 0.39	3.90 ± 0.25	5.12 ± 0.51	4.00 ± 0.23	5.53 ± 0.95	4.06 ± 0.41
PressLight	4.94 ± 0.33	4.85 ± 0.35	6.74 ± 0.95	5.30 ± 0.41	6.49 ± 1.07	4.98 ± 0.46
DQN	5.39 ± 0.36	5.04 ± 0.31	8.22 ± 1.29	6.60 ± 0.47	10.77 ± 2.91	6.77 ± 0.75
Maxpressure	9.07 ± 0.95	6.47 ± 0.37	8.89 ± 0.96	6.50 ± 0.45	8.88 ± 0.97	6.54 ± 0.47

[a]The percentages here refer to the CV penetration rates.

Table 8
Traffic demands.

Stages	Road network	Demand patterns	Arrival rate (vehicles/h/lane)			
			Arterial (E-W)	Arterial (W-E)	Side-street (N-S)	Side-street (S-N)
Pre-train	2-2	Fixed	300	300	150	150
Training	5-5	Dynamic	500-700-1000[a]	250-350-500	250-350-500	125-175-250

[a]The series of numbers represents the demands in different stages in the dynamic traffic demand. For example, 500-700-1000 means the traffic demand starts from 500 veh/h/lane, increases to 700 veh/h/lane, and ends with 1000 veh/h/lane.



(a) 2-2 road network for pre-train (b) 5-5 road network for training

Fig. 11. Illustration of traffic demand patterns for pre-train and training.

structures, and continue the training until convergence. Under the 10% or the 30% penetration rate, CVLight agents are pre-trained on the 2-2 road network with a fixed traffic demand pattern for 12,000 iterations, and then are loaded and trained for another 12,000 iterations on the 5-5 road network with a dynamic traffic demand pattern. To better demonstrate the effectiveness of the pre-train technique, we design traffic demand patterns that are significantly different between the pre-train stage and the training stage, as summarized in Table 8 and Fig. 11. For Table 8, the first column gives the stages, the second column presents the road network, the third column indicates the traffic demand patterns, and the fourth to sixth columns detail the arrival rate in each direction. Such traffic patterns are also illustrated in Fig. 11 where arrows represent the directions of arriving vehicles.

Comparison on convergence speed. We test the performance of the CVLight (Asym) agents under the 5-5 road network after every 2000 iterations in training with or without the pre-train technique and the result is detailed in Fig. 12. For Fig. 12, the x-axis is the iteration index during training and the y-axis represents the average delay per vehicle per intersection in a logarithmic scale. The blue line is the performance of CVLight (Asym) with the pre-train technique and the orange line is the performance of CVLight

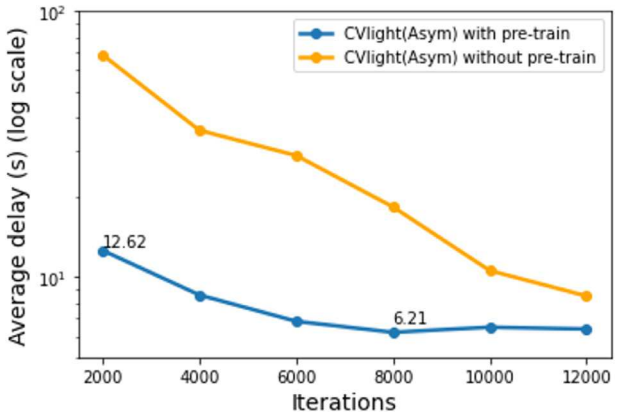


Fig. 12. Comparison between CVLight (Asym) with and without pre-train in the convergence speed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 9
Performance comparison between CVLight (Asym) with pre-train and baselines in the 5-5 network.

Algorithms	CV penetration rate	
	10%	30%
CVLight (Asym) with pre-train	**6.21** ± 0.62	**5.43** ± 0.85
Presslight	6.83 ± 0.55	5.54 ± 0.68
Maxpressure	7.85 ± 0.95	6.32 ± 0.20
DQN	8.70 ± 0.64	5.96 ± 0.34
Websters	11.02 ± 1.84	7.73 ± 1.43

(Asym) without the pre-train technique. The numbers above the blue line are the average delays of CVLight (Asym) with pre-train at corresponding iterations. From the result, we can see CVLight (Asym) with pre-train can converge in a much faster way than that without using the pre-train technique does. We train our models on one c5a.xlarge AWS EC2 instance with 16 vCPUs and 32 GB memory. The processors are the 2nd generation 3.3 GHz AMD EPYC 7002 series. By using the pre-train technique, the model begins to converge at the 6000th iteration, which is a 50% improvement compared to the total 12 000 iterations.

Comparison on performance under 5-5. Under the 5-5 road network with 10% and 30% penetration rates, the performance comparison between CVLight and baseline models is presented in Table 9. The first column of Table 9 lists the algorithms and the second to third columns show the average delay as well as the standard deviation of each algorithm under the 10% and 30% penetration rates, respectively. From Table 9, CVLight (Asym) with pre-train can notably outperform all benchmark algorithms, which demonstrates the good scalability of CVLight.

4. Case study on real-world intersections

In addition to synthetic intersections, we conduct experiments on a real-world road network of 4 intersections. In this section, we first introduce the new experiment set-up in Section 4.1 and then present experiment results in Section 4.2.

4.1. Environment set-up

Fig. 13 illustrates the intersection structure, traffic patterns, and traffic signal phase setting for the case study environment. Fig. 13(a) shows the road network structure and traffic flows of a real-world 2-by-2 network located in the intersections of Beaver Avenue and College Avenue from Atherton Street to Burrowes Street in State College, Pennsylvania, USA. The Beaver Avenue and College Avenue are arterials with unidirectional traffic; the Atherton Street and the Burrowes Street are side-streets with bidirectional traffic. For a better comparison with PressLight (Wei et al., 2019), we select this real-world road network because PressLight also gets tested under the corridor on the Beaver Avenue in the same area. Fig. 13(b) details the traffic signal phase setting for each intersection in the road network. Each row represents the signal timing plan for one intersection and each cell inside one row indicates one green phase, consisting of traffic movements that are allowed for that phase.

Vehicles' inter-arrival times follow a binomial distribution, with the same dynamic demand pattern shown in Table 4 and Fig. 6. Both the left-turn proportion and right-turn proportion are set as 10%. In this section, we focus on the performance comparison of CVLight (Asym) and other benchmark algorithms, considering the superior performance of CVLight (Asym) in Section 3.

(a) Beaver Ave. and College Ave. from Atherton St. to Burrowes St., Pennsylvania, USA: a real-world 2-by-2 road network

Intersection	Traffic Signal Phases	
1	2 1	3
2	6 4	5
3	8	7
4	9 11	10

(b) Traffic signal phases settings

Fig. 13. Illustrations of intersection structure, traffic movements, and signal phase setting in the real-world road network.

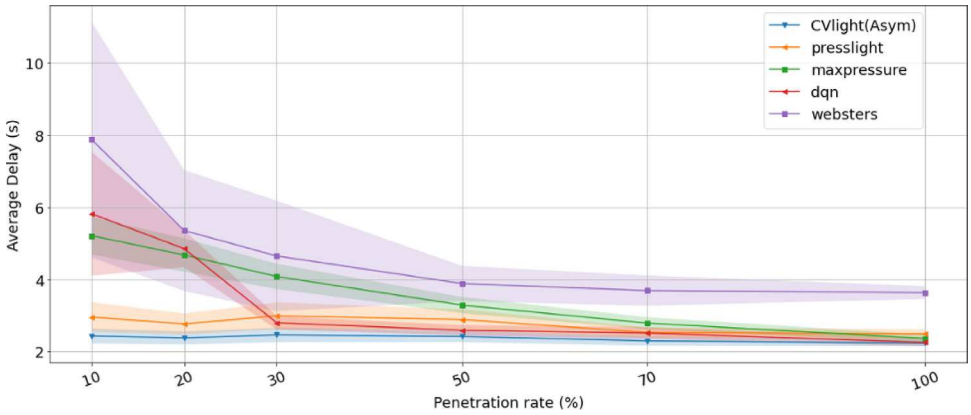


Fig. 14. Performance comparison under a real-world 2-by-2 road network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Result analysis

Fig. 14 illustrates the result of the performance comparison between CVLight (Asym) and baselines. The x-axis is the CV penetration rate and the y-axis represents the average delay per vehicle per intersection. The blue, orange, green, red, and purple lines represent the performance of CVLight (Asym), PressLight, Maxpressure, DQN, and Webster's, respectively. From Fig. 14, CVLight (Asym) can outperform all other benchmark algorithms across all penetration rates.

5. Conclusions and future research

This paper develops the CVLight model that learns with information from CVs and non-CVs and executes with data collected from CVs only. The model is characterized by the state and reward design that facilitates coordination among agents and utilizes travel delay and phase duration. We propose the Asym-A2C that takes advantage of the structure of the actor-critic algorithm so as to include non-CV information into the training process. Results of extensive experiments under various traffic demands, penetration

rates, as well as road networks, demonstrate the superiority of CVLight over other state-of-the-art benchmark algorithms and its good generalizability in penetration rates and traffic demands. By visualizing the learned policies of CVLight agents and the relative importance of input layer neurons in the critic neural network, we show how the Asym-A2C algorithm and our state design contribute to the good performance of CVLight (Asym), especially under low penetration rate scenarios. Moreover, the scalability of CVLight is further improved through the pre-train technique, which significantly decreases the training time of CVLight under a 5-by-5 road network.

Future work is threefold: (1) We plan to explore more state-of-the-art neural network interpretability methods to better interpret the learned policies of agents. (2) We will further improve the scalability of CVLight by introducing other deep learning methods, such as parameter sharing. (3) Scenarios like extremely low CV penetration can still be challenging for TSC. Following the suit of some existing studies, we will combine state estimation methods, such as queue estimation, with RL-TSC so as to further improve our model performance.

CRediT authorship contribution statement

Zhaobin Mo: Conceptualization, Formal analysis, Methodology, Visualization, Writing – original draft, Writing – review & editing. **Wangzhi Li:** Conceptualization, Formal analysis, Methodology, Visualization, Writing – original draft. **Yongjie Fu:** Formal analysis, Methodology, Visualization, Writing – review & editing. **Kangrui Ruan:** Formal analysis, Methodology, Visualization, Writing – review & editing. **Xuan Di:** Conceptualization, Methodology, Supervision, Writing – review & editing.

Acknowledgments

This work is partially sponsored by the National Science Foundation (NSF), United States under the CAREER award CMMI-1943998 and NSF CPS-2038984. We also thank Mr. Ujwal Dinesha and Yaxing Cai for assisting in coding in the first phase of this project.

References

Abboud, K., Omar, H.A., Zhuang, W., 2016. Interworking of DSRC and cellular network technologies for V2X communications: A survey. IEEE Trans. Veh. Technol. 65 (12), 9457–9470.

Agiwal, M., Roy, A., Saxena, N., 2016. Next generation 5G wireless networks: A comprehensive survey. IEEE Commun. Surv. Tutor. 18 (3), 1617–1655.

Al Islam, S.B., Aziz, H.A., Wang, H., Young, S.E., 2018. Minimizing energy consumption from connected signalized intersections by reinforcement learning. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 1870–1875.

Al Islam, S.B., Hajbabaie, A., Aziz, H.A., 2020. A real-time network-level traffic signal control methodology with partial connected vehicle information. Transp. Res. C 121, 102830.

Aziz, H., Wang, H., Young, S., et al., 2019. Investigating the Impact of Connected Vehicle Market Share on the Performance of Reinforcement-Learning Based Traffic Signal Control. Technical Report, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).

Beak, B., Head, K.L., Feng, Y., 2017. Adaptive coordination based on connected vehicle technology. Transp. Res. Rec. 2619 (1), 1–12.

Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D., 2011. SUMO–Simulation of urban mobility: an overview. In: Proceedings of SIMUL 2011, the Third International Conference on Advances in System Simulation. ThinkMind.

Chacha Chen, H.W., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., Li, Z., 2020. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In: Proceeding of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI'20). New York, NY.

Chettri, L., Bera, R., 2020. A comprehensive survey on internet of things (IoT) toward 5G wireless systems. IEEE Internet Things J. 7 (1), 16–32.

Chu, T., Wang, J., Codecà, L., Li, Z., 2019. Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans. Intell. Transp. Syst. 21 (3), 1086–1095.

Di, X., Shi, R., 2021. A survey on autonomous vehicle control in the era of mixed-autonomy: From physics-based to AI-guided driving policy learning. Transp. Res. C 125, 103008.

Feng, Y., Head, K.L., Khoshmagham, S., Zamanipour, M., 2015. A real-time adaptive signal control in a connected vehicle environment. Transp. Res. C 55, 460–473.

Feng, Y., Zamanipour, M., Head, K.L., Khoshmagham, S., 2016. Connected vehicle–based adaptive signal control and applications. Transp. Res. Rec. 2558 (1), 11–19.

Feng, Y., Zheng, J., Liu, H.X., 2018. Real-time detector-free adaptive signal control with low penetration of connected vehicles. Transp. Res. Rec. 2672 (18), 35–44.

Gao, K., Han, F., Dong, P., Xiong, N., Du, R., 2019. Connected vehicle as a mobile sensor for real time queue length at signalized intersections. Sensors 19 (9), 2059.

Genders, W., Razavi, S., 2019. An open-source framework for adaptive traffic signal control. arXiv preprint arXiv:1909.00395.

Gevrey, M., Dimopoulos, I., Lek, S., 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecol. Model. 160 (3), 249–264.

Gong, Y., Abdel-Aty, M., Cai, Q., Rahman, M.S., 2019. Decentralized network level adaptive signal control by multi-agent deep reinforcement learning. Transp. Res. Interdiscip. Perspect. 1, 100020.

Goodall, N.J., Park, B., Smith, B.L., 2014. Microscopic estimation of arterial vehicle positions in a low-penetration-rate connected vehicle environment. J. Transp. Eng. 140 (10), 04014047.

Goodall, N.J., Smith, B.L., Park, B., 2013. Traffic signal control with connected vehicles. Transp. Res. Rec. 2381 (1), 65–72.

Guo, Q., Li, L., Ban, X.J., 2019. Urban traffic signal control with connected and automated vehicles: A survey. Transp. Res. C 101, 313–334.

Hao, P., Ban, X., 2015. Long queue estimation for signalized intersections using mobile data. Transp. Res. B 82, 54–73.

He, Q., Head, K.L., Ding, J., 2012. PAMSCOD: PLatoon-based arterial multi-modal signal control with online data. Transp. Res. C 20 (1), 164–184.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034.

Hu, H.-C., Smith, S.F., Goldstein, R., 2019. Cooperative schedule-driven intersection control with connected and autonomous vehicles. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 1668–1673.

Hussain, A., Wang, T., Jiahua, C., 2020. Optimizing traffic lights with multi-agent deep reinforcement learning and V2X communication. arXiv preprint arXiv:2002.09853.
Khurpade, J.M., Rao, D., Sanghavi, P.D., 2018. A survey on IOT and 5G network. In: 2018 International Conference on Smart City and Emerging Technology (ICSCET). pp. 1–3.
Kim, J., Jung, S., Kim, K., Lee, S., 2019. The real-time traffic signal control system for the minimum emission using reinforcement learning in V2X environment. Chem. Eng. Trans. 72, 91–96.
Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L., 2012. Recent development and applications of SUMO-simulation of urban mobility. Int. J. Adv. Syst. Meas. 5 (3&4).
Lee, J., Park, B., Yun, I., 2013. Cumulative travel-time responsive real-time intersection control algorithm in the connected vehicle environment. J. Transp. Eng. 139 (10), 1020–1029.
Li, W., Ban, X., 2018. Connected vehicles based traffic signal timing optimization. IEEE Trans. Intell. Transp. Syst. 20 (12), 4354–4366.
Li, W., Ban, X., 2020. Connected vehicle-based traffic signal coordination. Engineering 6 (12), 1463–1472.
Li, W., Ban, X.J., Wang, J., 2016. Traffic signal timing optimization incorporating individual vehicle fuel consumption characteristics under connected vehicles environment. In: 2016 International Conference on Connected Vehicles and Expo (ICCVE). IEEE, pp. 13–18.
Li, L., Okoth, V., Jabari, S.E., 2020. Backpressure control with estimated queue lengths for urban network traffic. arXiv preprint arXiv:2006.15549.
Li, L., Wen, D., Yao, D., 2013. A survey of traffic control with vehicular communications. IEEE Trans. Intell. Transp. Syst. 15 (1), 425–432.
Liu, W., Liu, J., Peng, J., Zhu, Z., 2014. Cooperative multi-agent traffic signal control system using fast gradient-descent function approximation for V2I networks. In: 2014 IEEE International Conference on Communications (ICC). IEEE, pp. 2562–2567.
Liu, W., Qin, G., He, Y., Jiang, F., 2017. Distributed cooperative reinforcement learning-based traffic signal control that integrates v2x networks' dynamic clustering. IEEE Trans. Veh. Technol. 66 (10), 8667–8681.
Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., ner, E.W., 2018. Microscopic traffic simulation using SUMO. In: The 21st IEEE International Conference on Intelligent Transportation Systems. IEEE, URL: https://elib.dlr.de/124092/,
Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR, pp. 1928–1937.
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., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.
Mohebifard, R., Al Islam, S.B., Hajbabaie, A., 2019. Cooperative traffic signal and perimeter control in semi-connected urban-street networks. Transp. Res. C 104, 408–427.
Mohebifard, R., Hajbabaie, A., 2018. Real-Time Adaptive Traffic Metering in a Connected Urban Street Network. Technical Report.
Van der Pol, E., Oliehoek, F.A., 2016. Coordinated deep reinforcement learners for traffic light control. In: Proceedings of Learning, Inference and Control of Multi-Agent Systems (At NIPS 2016).
Priemer, C., Friedrich, B., 2009. A decentralized adaptive traffic signal control using V2I communication data. In: 2009 12th International IEEE Conference on Intelligent Transportation Systems. IEEE, pp. 1–6.
SAS, 2015. The connected vehicle: Big data, big opportunities. https://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/connected-vehicle-107832.pdf [Online; accessed 03.31.2021].
Shou, Z., Chen, X., Fu, Y., Di, X., 2022. Multi-agent reinforcement learning for markov routing games: a new modeling paradigm for dynamic traffic assignment. Transportation Research Part C: Emerging Technologies 137, 103560.
Shou, Z., Di, X., 2020. Reward design for driver repositioning using multi-agent reinforcement learning. Transp. Res. C 119 (102738).
Sutton, R.S., Barto, A.G., et al., 1998. Introduction to Reinforcement Learning, Vol. 135. MIT press Cambridge.
Tiaprasert, K., Zhang, Y., Wang, X.B., Zeng, X., 2015. Queue length estimation using connected vehicle technology for adaptive signal control. IEEE Trans. Intell. Transp. Syst. 16 (4), 2129–2140.
USDOT, 2019. Connected vehicle. https://its.dot.gov/research_areas/WhitePaper_connected_vehicle.htm [Online; accessed 03.31.2021].
Varaiya, P., 2013. Max pressure control of a network of signalized intersections. Transp. Res. C 36, 177–195.
Webster, F.V., 1958. Traffic signal settings. Technical Report.
Wei, H., Chen, C., Zheng, G., Wu, K., Gayah, V., Xu, K., Li, Z., 2019. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1290–1298.
Wu, T., Zhou, P., Liu, K., Yuan, Y., Wang, X., Huang, H., Wu, D.O., 2020. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. IEEE Trans. Veh. Technol..
Yan, S., Zhang, J., Büscher, D., Burgard, W., Efficiency and equity are both essential: a generalized traffic signal controller with deep reinforcement learning. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 5526–5533.
Yang, K., Menendez, M., 2018. Queue estimation in a connected vehicle environment: A convex approach. IEEE Trans. Intell. Transp. Syst. 20 (7), 2480–2496.
Yang, S., Yang, B., Wong, H.-S., Kang, Z., 2019. Cooperative traffic signal control using multi-step return and off-policy asynchronous advantage actor-critic graph algorithm. Knowl.-Based Syst. 183, 104855.
Zhang, R., Ishikawa, A., Wang, W., Striner, B., Tonguz, O.K., 2020. Using reinforcement learning with partial vehicle detection for intelligent traffic signal control. IEEE Trans. Intell. Transp. Syst..
Zheng, J., Liu, H.X., 2017. Estimating traffic volumes for signalized intersections using connected vehicle data. Transp. Res. C 79, 347–362.