Traffic Signal Control Using Offline Reinforcement Learning

Xingyuan Dai[1,2], Chen Zhao[1,2], Xiaoshuang Li[1,2], Xiao Wang[2], Fei-Yue Wang[2]
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]The State Key Laboratory for Management and Control of Complex Systems,
Institute of Automation, Chinese Academy of Sciences
{daixingyuan2015, zhaochen2020, lixiaoshuang2017, x.wang, feiyue.wang}@ia.ac.cn

***Abstract*—The problem of traffic signal control is essential but remains unsolved. Some researchers use online reinforcement learning, including the off-policy one, to derive an optimal control policy through interaction between agents and environments in simulation. However, it is difficult to deploy the policy in real transportation systems due to the gap between simulated and real traffic data. In this paper, we consider an offline reinforcement learning method to tackle the problem. First, we construct a realistic traffic environment and obtain offline data based on a classic actuated traffic signal controller. Then, we use an offline reinforcement learning algorithm, namely conservative Q-learning, to learn an efficient control policy via offline datasets. We conduct experiments on a typical road intersection and compare the conservative Q-learning policy with the actuated policy and two data-driven policies based on off-policy reinforcement learning and imitation learning. Empirical results indicate that in the offline-learning setting the conservative Q-learning policy performs significantly better than other baselines, including the actuated policy, but the other two data-driven policies perform poorly in test scenarios.**

***Index Terms*—Traffic signal control, offline reinforcement learning, off-policy reinforcement learning.**

I. Introduction

Urban traffic control plays a vital role in maintaining efficiency and safety for intelligent transportation systems [1]. As an essential part of urban traffic control, traffic signal controllers direct traffic flows at road intersections by determining traffic signals' phase and duration time. Traffic signal control aims to derive a control policy for traffic signals that minimizes the delay time for all vehicles crossing the intersection. To achieve this goal, researchers have proposed various traffic signal control methods. Initially, the traffic signal control method uses a kind of fixed-time policy. In such a policy, the fixed phase sequence and corresponding duration time of the traffic signal are manually predetermined by experienced experts according to their observation for historical traffic flows around the intersection. The policy can relieve the traffic jam, but the predetermined fixed-time controllers have difficulties adapting to variant traffic patterns. The drawbacks provoke researchers to turn to actuated traffic signal controllers to handle different traffic demands adaptively.

The actuated controller can adjust its timing plans to adapt to the change of traffic flow, which is detected by sensors placed in the road network [2]. A typical actuated controller implements a kind of greedy policy, keeping the current phase until detecting a sufficient gap between two continuous vehicles and then switching to the next phase. Due to the ability to adapt to different traffic patterns, actuated controllers outperform fixed-time controllers to reduce traffic delay in most scenarios. Currently, actuated controllers have been incorporated in some advanced commercial traffic signal control systems like SCATS [3] and SCOOT [4], which are successfully deployed in hundreds of cities worldwide. Although actuated controllers have been successfully applied in practice, they are challenging to guarantee an optimal policy for regional traffic control since a primary actuated controller focuses on an isolated intersection without considering the impact of upstream and downstream intersections [5].

To obtain an optimal global policy for regional traffic control, Fei-Yue Wang proposed the concept of parallel transportation systems in [1]. In parallel transportation systems, the artificial systems constructed from the real systems are used for data-driven policy optimization via a global target. Most importantly, the policy learned from artificial systems should guide the operation of the real system. Along with this idea, researchers have proposed various data-driven methods for traffic signal control. However, most of the works are based on simulation rather than artificial systems since these methods do not consider the connection between virtual and real systems. Nonetheless, we will indiscriminately refer to the virtual systems used for interaction to optimize the control strategies as simulated systems for ease of introduction.

For simulation-based data-driven traffic signal control, the reinforcement learning (RL) based method is a representation. Generally, RL approaches first construct a simulation system corresponding to the real one and then learn the control policy from scratch by interacting with the simulated traffic environment to maximize the notion of cumulative reward [6]. To deploy the RL policy in real scenarios, researchers have delved into the studies that bridge the gap between real and simulated transportation systems. An intuitive approach is to build a simulation system that is as real as possible while covering most traffic patterns in real systems. The simulation system can be used to train a robust policy that

This work was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B090921003, in part by the National Natural Science Foundation of China under Grant U1811463, and in part by the Intel Collaborative Research Institute for Intelligent and Automated Connected Vehicles (ICRI-IACV).

is then transferred to the real traffic signal controller [7], [8]. Nevertheless, we find it hard to believe the policy trained from the simulation is qualified for real systems. The deployment process still needs many human resources to monitor the operation conditions carefully and deal with emergencies.

Another widely used data-driven approach, imitation learning, can directly learn a control policy from real-world traffic control datasets [9]. However, imitation learning methods, which model the mapping from traffic states to control actions, opt to copy the policy they learned and may overfit training datasets. The overfitting is harmful to the controller's robustness, leading to poor decision-making on unseen scenarios.

This paper tries to bridge the gap between optimization and deployment for traffic signal control policies. For this, we build a realistic traffic environment with classic actuated signal controllers by default as the real scenario, which can be used for data collection and control policy deployment but not for interaction. We first want to optimize the control policy via offline data collected from the realistic environment rather than interaction with simulated environments. The optimization scheme will alleviate the domain difference between training and testing scenarios for traffic signal control and enable the learned control policy to deploy in real transportation systems reassuringly. Furthermore, we hope the optimized control policy can match or even exceed the real controller's policy it learns on generalized scenarios.

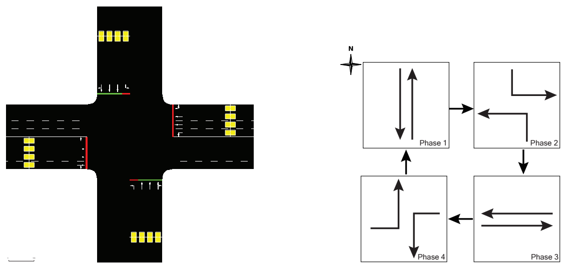
To achieve the two goals above, we consider an offline RL model to optimize traffic signal control policies. First, we build an offline dataset from realistic traffic environments with actuated signal controllers; all elements in the dataset can be collected from real transportation systems. Then, we use an efficient offline RL model, namely conservative Q-learning (CQL) [10], to optimize the control policy based on the offline dataset. The main contributions of this paper are as follows:

1) We rethink a critical problem in data-driven traffic signal control about trade-offs between the optimality in policy optimization and reliability in policy deployment.
2) We explore an effective workflow to learn a trusty offline RL-based traffic signal control policy for deployment without interaction with the environment.
3) The offline RL-based traffic signal control policy can outperform the rule-based behavior policy and two typical data-driven strategies, i.e., imitation learning and off-policy RL, in the offline learning setting.

To explain our findings further, we organize the rest of the paper as follows. Section II introduces the problem formulation of traffic signal control. Section III introduces the implementation details for offline RL-based traffic signal control. Section IV compares the offline RL controller with four baselines. Finally, Section V concludes the paper.

II. Problem Formulation

In this section, we will formulate the traffic signal control problem as a Markov decision process (MDP), which will be solved by the offline RL method introduced in Section III.



(a) (b)

Fig. 1: The studied realistic traffic environment, which can be used for data collection and control policy deployment but not for interaction like the simulated environment. (a) A typical signalized intersection with an actuated signal controller by default. (b) The signal phase sequence.

The formulation of MDP in this paper is based on a constructed realistic traffic environment with a single intersection as shown in Fig. 1, and the following analysis can be easily generalized to other scenarios. Note that the realistic environment cannot be used to conduct online learning like the simulated environment; we can only collect offline data from realistic environments with the default actuated policy and deploy the offline-learning policy for evaluation. Although environment settings discussed in this paper are different from that in prevalent RL-based traffic signal control, the control problem can still be formulated as a standard MDP.

The MDP is a formulation of sequential decision-making, where actions taken by the agent at each time step will affect both immediate and subsequent rewards [11]. A typical MDP contains five elements $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ denotes the set of actions, $\mathcal{P}\colon \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ denotes the transition probability from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ given action $a \in \mathcal{A}$, $\mathcal{R}\colon \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function determining immediate rewards received by the agent for a transition from (s, a) to s', and γ is the discount factor.

At each time step t, the traffic signal controller at the intersection executes an action a_t that determines the timing plan for the traffic signal according to the traffic state s_t to ensure that vehicles move as speedily, smoothly, and safely as possible. The traffic environment will transition to the next state s_{t+1} and return a reward r_t as evaluation of the action a_t. In this paper, we define Δt as the control period for the RL agent so that the environment runs for Δt after each MDP time step. Then, we will detail settings of states, actions, and rewards for traffic signal control and formulate the problem.

State: The state $s_t \in \mathcal{S}$ represents the accessible information from the environment by the RL agent at time t. The data are generally collected by traffic sensors distributed in the road network. To make our experimental settings more realistic, we only consider the traffic data accessible in real traffic sensors and define the state at time t as

$$s_t = \left\{ \left\{ q_{t,l}, v_{t,l} \right\}_{l \in L}, p_t, d_t \right\} \tag{1}$$

where L denotes the set of incoming lanes of the intersection,

$q_{t,l}$ and $v_{t,l}$ denote the queue length and the total number of approaching vehicles along each incoming lane l of the intersection, p_t is the one-hot encoding for the current phase, and d_t is its spent duration.

Action: The action $a_t \in \mathcal{A}$ is taken by the RL agent at time t according to the state and its policy. Considering the practicability, we choose the phase switch as the controlled object. Specifically, a feasible and complete phase sequence for the intersection is predefined as shown in Fig. 1b, and the agent selects keeping the current phase or switching to the next phase for a duration of Δt at each time step. Note that there is t_y ($t_y < \Delta t$) yellow time when switching the phase to guarantee traffic safety in the intersection.

Reward: The reward r_t is the feedback from the environment by the reward function $r(s_t, a_t)$, representing a performance indicator to measure the action taken by the agent. In this paper, we use *pressure* [12] of the intersection as reward indicators. The motivation for such a choice is that it is demonstrated that minimizing pressure is equivalent to minimizing average travel time, which is the main optimizing goal in traffic signal control but is hard to obtain in reality and optimize directly [13]. The pressure of the intersection is defined as the value of the total number of vehicles entering lanes minus the total number of leaving vehicles and can be denoted as

$$P_t = \sum_{l \in L} v_{t,l} - \sum_{o \in O} v_{t,o}, \tag{2}$$

where L and O denote the set of incoming lanes and outgoing lanes of the intersection and $v_{t,l}$ and $v_{t,o}$ represent vehicle numbers in the incoming lane l and outgoing lane o at time step t. As the environment responds to the action a_t at time $t+1$, accompanied by the pressure evaluation, we define the reward as

$$r_t = -P_{t+1}. \tag{3}$$

Based on the definition above, we can formulate traffic signal control as the following MDP problem.

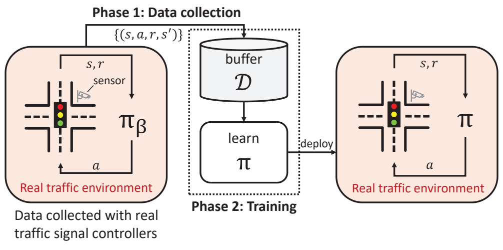
***Problem* 1:** Given a single-intersection traffic environment, we search for a signal control policy $\pi = \{\pi \colon \mathcal{S} \mapsto \mathcal{A}\}$ for the intersection agent to maximize the expected value of cumulative reward

$$G := \mathbb{E}\left[\sum_{t \geq 0} r_t \middle| a_t \sim \pi\left(\cdot \mid s_t\right)\right]. \tag{4}$$

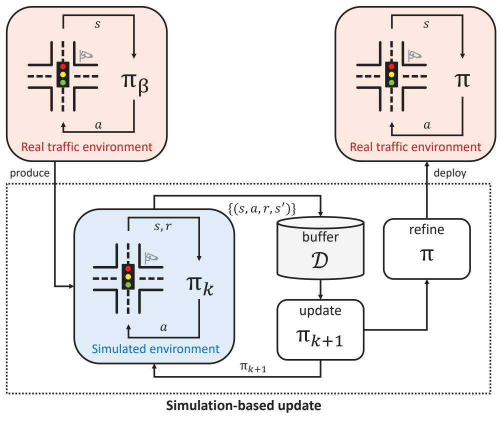
III. Methodology

A. Offline RL-based Traffic Signal Control

As illustrated in Fig. 2a, the workflow of offline RL-based traffic signal control contains two successive phases: offline traffic data collection and control policy optimization. In the data collection phase, we collect road information and the signal control behavior based on the policy π_β within a period from the real traffic environment, and structure them as the transition (s, a, r, s') to build offline datasets. In the offline training phase, the offline data are fed to a buffer, which



(a) Offline RL-based traffic signal control



(b) Online RL-based traffic signal control

Fig. 2: The general workflow of offline RL-based (a) and online RL-based (b) traffic signal control. In offline RL-based traffic signal control (a), the control policy is learned by data that can be directly collected from the real traffic environment with real signal controllers like the actuated one. In online RL-based traffic signal control (b), the control policy should be learned in a simulated environment and deal with gaps between the real and simulated scenarios.

samples data to optimize the control policy π. The policy π is promising for deployment in the real traffic environment since it is learned from real-world data.

The procedure for offline RL-based traffic signal control is different from online RL-based, which is illustrated in Fig. 2b. For online RL-based traffic signal control, we need first to build a simulated traffic environment corresponding to the real one and then enable the agent to optimize the control policy by interaction with the simulated environment. Finally, we should refine the learned policy to adapt to the real traffic environment before deployment. The modeling error introduced by each procedure can be mitigated by high-level learning frameworks like parallel learning [14]. On the contrary, offline RL-based traffic signal control does not need to rely on simulated environments. So it will reduce environmental modeling procedures and mainly focus on learning methods for better generalizability.

To achieve better generalizability, let us elaborate the workflow of offline RL-based traffic signal control for the studied traffic scenario. As shown in Fig. 2a, in the data collection phase, we first gather the raw data containing lane-level queue length and approaching vehicles from road sensors, and the signal phase index as well as its spent duration time from the signal controller. These elements collected at time step t are formulated as the state s_t. Then, the signal control behavior at time step t describing whether the signal controllers keep the current phase or switch to the next phase is recorded as the action a_t. Apart from states and actions, we need to construct rewards as critical components for offline reinforcement learning. The reward function is defined by traffic pressure in the intersection, as described in Section II. So we can easily calculate the reward r_t using the collected data through Eqs. (2) and (3). After constructing the offline datasets, we can use observed transitions (s_t, a_t, r_t, s_{t+1}) sampled from the dataset to optimize the signal control policy. In the training phase, we use an efficient offline RL method called CQL, in which a conservative Q-function is learned to give rise to the policy π for traffic signal control. The detailed explanation of CQL will be presented in Subsection III-C.

The motivation for applying CQL to real traffic signal control is that the method has overwhelming advantages compared with two representative data-driven optimization methods available for offline datasets, i.e., imitation learning and off-policy RL methods. First, CQL methods enable the agent to learn a better policy than the behavior policy by taking advantage of dynamic programming, which could infer potentially optimal behavior from suboptimal parts of trajectories in datasets [10]. Contrarily, general imitation learning methods like behavioral cloning only learn the behavior policy from the datasets and may not generalize well to unseen scenarios. Second, CQL performs conservative estimation for Q-function to deal with the distributional shift between the behavior policy that collected data and the learned policy [10]. However, the off-policy RL method like Q-learning designed for online learning is challenging to handle, leading to a poor control policy for generalizability.

By performing CQL on a previously collected real traffic dataset, we could learn an effective and well-generalized policy for real traffic signal control. The property of trustworthy policy learning for CQL derives from its rigorous theoretical derivation for safe policy improvement. Before presenting the formulation and details of CQL, we will first introduce Q-learning, which is the fundamental of CQL.

B. Q-learning

As a fundamental RL method, Q-learning aims to learn a policy that maximizes the long-term return of MDP by approximating the optimal action-value function Q^*. For an infinite MDP, the action-value function, also known as Q-function for a policy, is defined as the expected return starting from the current state s_t with action a_t under the policy π:

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \right] \tag{5}$$

where γ is the discount parameter. By using the Bellman equation [11], we can decompose the Q-function as

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[r_t + \gamma Q_\pi(s_{t+1}, a_{t+1}) \right]. \tag{6}$$

The Q-learning theory [11] guarantees that the Q-function will converge to the optimal action-value function Q^* by iterative transition data collection and update

$$\begin{aligned} Q(s_t, a_t) \leftarrow & Q(s_t, a_t) \\ & + \beta \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right] \end{aligned} \tag{7}$$

where β is the learning rate.

Empirically, to stabilize the training process, we often use a separate target Q-function $\bar{Q}$, which is periodically updated to the learned Q-function. At each iteration, we collect an online dataset $\mathcal{D}_o$ and train the Q-function by minimizing temporal difference (TD) error

$$\min_Q \mathbb{E}_{s,a,s' \sim \mathcal{D}_o} \left[\left(r(s,a) + \gamma \max_{a'} \bar{Q}(s', a') - Q(s,a) \right)^2 \right]. \tag{8}$$

Note that Q-learning is designed for online learning in which data are collected through the interaction between agents and environments. This enables the learned Q-function to accurately estimate action values since it is trained on actions sampled from its relative behavior policy. However, in offline learning, datasets that are collected under the predefined behavior policy may be irrelevant to the learned policy. This will lead to the problem of action distribution shift [10], [15] during training and make the Q-function erroneously give high action-values for unseen actions outside the dataset. The problem is serious for offline RL-based traffic signal control since behavior policies like actuated ones for real signal controllers are rule-based, which are different from data-driven policies. To better tackle the problem of overestimation in Q-learning, we consider conservative Q-learning for traffic signal control.

C. Conservative Q-learning

CQL adds regularizers for the value function and the learned policy on top of off-policy RL algorithms like standard Q-learning. So we can leverage some properties of Q-learning to formulate the optimization function of CQL.

Suppose we have a previously collected dataset $\mathcal{D}$ containing transitions (s, a, r, s') obtained by the behavior policy π_β, which is the actuated policy in this paper. We want to learn an effective policy π from the offline dataset without intersection. To optimize the policy, we need to estimate its value $V_\pi(s)$, which indicates the expected return starting from the current state s alongside the policy π. To prevent the overestimation of the policy value $V_\pi(s)$, CQL proposes a lower-bound Q-function $Q_\pi(s)$ by which we can obtain the lower-bound value $V_\pi(s) = \mathbb{E}_{\pi(a|s)}[Q_\pi(s,a)]$. Moreover, the Q-function can be used to obtain the policy π, optimized to approximate the one that maximizes the Q-function.

The general form of optimization function for CQL is

$$\begin{aligned} & \min_Q \max_\pi \alpha \left[\mathbb{E}_{s\sim\mathcal{D}, a\sim\pi}[Q(s,a)] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)] + \mathcal{R}(\pi) \right] \\ & + \frac{1}{2} \mathbb{E}_{s,a,s'\sim\mathcal{D}} \left[\left(r(s,a) + \gamma \max_{a'} \bar{Q}(s', a') - Q(s,a) \right)^2 \right], \end{aligned} \tag{9}$$

where the first and second items are CQL regularizers for the Q-function; the third item is the regularizer for the policy; and the remaining components comprise standard TD error, which is the optimization objective for Q-learning as shown in (8). In (9), the first item, the penalty of the Q-function under the distribution $\pi(a|s)$, obtains a conservative Q-function under which the estimated Q-value lower-bounds its true value. The second item, which maximizes Q-values under the data distribution $\pi_\beta(a|s)$ is used to attain a tighter lower bound for $V^\pi(s)$. The last regularizer item can be defined as the entropy of π, i.e., $\mathcal{H}(\pi)$, to better estimate Q-values [16].

By substituting $\mathcal{R}(\pi) = \mathcal{H}(\pi)$ in (9), and replacing π as $\arg\max_\pi \mathbb{E}_{s\sim\mathcal{D},a\sim\pi}[Q(s,a)] + \mathcal{H}(\pi)$, we can obtain the following optimization objective for the Q-function

$$\min_Q \alpha \mathbb{E}_{s\sim\mathcal{D}}\left[\log\sum_a \exp(Q(s,a)) - \mathbb{E}_{a\sim\pi_\beta(a|s)}[Q(s,a)]\right] + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(r(s,a) + \gamma\max_{a'}\bar{Q}(s',a') - Q(s,a)\right)^2\right]. \tag{10}$$

The objective enables us to train the Q-function by the offline dataset $\mathcal{D}$, and the trained Q-function can be leveraged to infer the action a given state s by a greedy policy, i.e., $a = \arg\max_{a\in\mathcal{A}} Q(s,a)$, during evaluation stages.

IV. Experiments

A. Datasets

In the experiment, we use SUMO [17] to build realistic traffic environments and evaluate the performance of different traffic signal control policies. SUMO can achieve fine-grained representations for elements in different traffic scenarios like emissions or individual vehicle routes as a microscopic traffic modeling tool that models each vehicle individually. So it meets our need for environmental realism.

As shown in Fig. 1, we build a realistic traffic environment with a single signalized intersection. The intersection has four arms with the same length of $750\,\mathrm{m}$ and the speed limit $13.89\,\mathrm{m/s}$. Each arm has eight two-way lanes, i.e., four incoming lanes and four outgoing lanes, and its incoming lanes include a left-turn lane. The traffic signal in the intersection contains four sequential phases: N-S straight phase, N-S left-turn phase, E-W straight phase, and E-W left-turn phase. Note that the right-turn phase is always allowed.

Apart from the road network and traffic signal settings, the traffic demand in this paper is considered to follow a uniform distribution; that is to say, traffic flows are generated uniformly throughout the runtime. Each runtime will last $3,600\,\mathrm{s}$, and a total of $7,200$ vehicles will be loaded into the environment. The turning rate of vehicles is set to 0.25. We generate different traffic demand files for a fair comparison. The demand file indicates departure time and driving routes for each vehicle in the runtime. We use 50 demand files, which are directed by the actuated traffic signal controller in the realistic environment, to generate offline datasets, and use 30 demand files to evaluate different control policies.

B. Control Policy Settings and Performance Criteria

This paper compares the CQL policy with four typical control policies; their details are as follows:

1) **Fixed-time policy:** The policy is widely used in real traffic environments. In our settings, the total duration for straight phases is $33\,\mathrm{s}$ and for left-turn phases is $6\,\mathrm{s}$.
2) **Actuated policy [2]:** The rule-based policy is widely deployed in real traffic environments. In our settings, the actuated traffic signal controller makes decisions every period of time Δt. It will switch to the next phase if detecting no vehicle via induction loop detectors related to the current phase within a time gap t_{gap}. In addition to being a baseline, the actuated policy is also used as the behavior policy to generate offline datasets for data-driven policy optimization.
3) **Behavioral cloning (BC) policy [9]:** The imitation learning policy tries to copy the behavior policy by learning a mapping function from states to actions based on state-action pairs sampled from the offline demonstration dataset.
4) **Deep Q-network (DQN) policy [6]:** The policy approximates the Q-function using a deep neural network, where Q-learning updates the Q-function via the transitions sampled from the offline dataset. Although Q-learning is designed for online learning, its off-policy character enables the agent to optimize policies according to objective (8) by offline learning.
5) **CQL policy [10]:** The offline RL policy approximates the Q-function using a deep neural network like DQN, but the learning process uses objective (10).

We evaluate the control performance through the total *output* criterion, which indicates the number of vehicles reaching destinations in a given environmental period:

$$O_{\text{total}} = \sum_{t=1}^{T} O_t. \tag{11}$$

Here, $T = 3,600$ in our experiments. Generally, higher total output means better performance for the control policy.

C. Implementation Details

In the experiment, we set the control period $\Delta t = 5\,\mathrm{s}$, and yellow time $t_y = 3\,\mathrm{s}$, so each environmental runtime will last for 720 MDP steps. The threshold of detection time gap t_{gap} for the actuated policy is $2\,\mathrm{s}$.

To stabilize training for three data-driven methods, we normalize states and rewards. The methods share the same network structure and common learning parameters. The neural network contains 2 hidden layers, each with 256 units. The optimization algorithm is Adam with minibatches of size 32, and the learning rate is 6.25×10^{-5}. Each data-driven policy is trained for $720,000$ iterations on offline datasets. For DQN and CQL policies, the target Q-network is updated every $20,000$ iterations. The weight α for CQL objective (10) is 0.01.

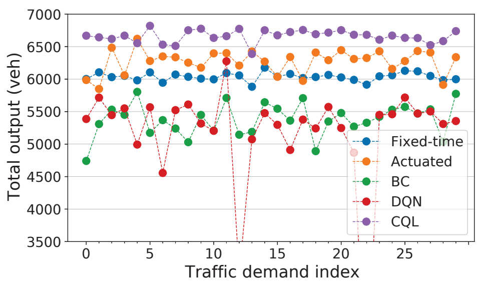


Fig. 3: Performance comparison of different control policies for 30 traffic demands.

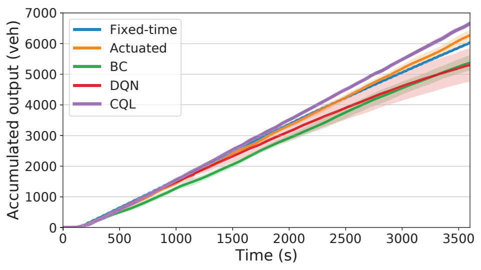


Fig. 4: Accumulated output flow along with time for different control policies. The solid line shows the average output for 30 traffic demands, and the shade shows the variance.

D. Performance Comparison

Fig. 3 plots total output on 30 traffic demands for different control policies. We can find that the CQL policy outperforms other baselines significantly, including the actuated policy it learns. The result indicates that CQL can learn a better policy for traffic signal control than the behavior policy like the actuated one while maintaining generalizability. On the contrary, BC and DQN policies, also trained with data from the actuated policy, perform poorly on the out-of-training traffic demands. Worse still, they deliver fewer cars to destinations compared with the fixed-time policy. We can also find that BC as a traditional supervised learning method has better robustness than the off-policy RL method DQN. This result may be induced by the overestimation in the offline-learning process for DQN, as discussed in Subsection III-A.

To better understand the experiment results, we show the accumulated output flow over the runtime for 30 tests in Fig. 4. We can find that CQL learns an effective policy that maintains the highest output rate (corresponding to the slope of the curve) throughout the episode. Compared with CQL, DQN without conservative penalty can only handle the traffic flow in the early stage of the episode but performs poorly later accompanied by high variance.

V. Conclusion

In this paper, with the help of the constructed realistic traffic environment, we explore the feasibility of learning a quantified RL-based traffic signal controller for real traffic scenarios with only offline data from the real world. We validate the effectiveness of workflow for offline RL-based traffic signal control by a typical offline RL method CQL. The CQL policy could attain comparable performance to the actuated policy and outperforms other data-driven methods by leveraging the data collected via the classic actuated policy.

Non-trivial future works are remaining for the real-world applications of the CQL model. Since CQL-like offline RL approaches have been demonstrated better at leveraging real-world data than behavioral cloning in many practical applications like robotics [10], it is possible to use such methods to learn better signal control policies from human experience or elaborate rules. Moreover, offline RL can be served as a module of computational experiments in parallel transportation systems [1]. The feature of offline policy optimization will bridge the gap between real and artificial systems.

References

[1] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 630–638, Sep. 2010.

[2] G. F. Newell, "Properties of vehicle-actuated signals: I. one-way streets," *Transportation Science*, vol. 3, no. 1, pp. 30–52, 1969.

[3] P. R. Lowrie, "SCATS, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic," *Roads and Traffic Authority NSW, Darlinghurst, NSW Australia*, 1990.

[4] P. B. Hunt, D. I. Robertson, R. D. Bretherton *et al.*, "The SCOOT on-line traffic signal optimisation technique," *Traffic Engineering & Control*, vol. 23, no. 4, Apr. 1982.

[5] P. G. Balaji, X. German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 177–188, 2010.

[6] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, Jul. 2016.

[7] A. Müller, V. Rangras, G. Schnittker *et al.*, "LemgoRL: An open-source benchmark tool to train reinforcement learning agents for traffic signal control in a real-world simulation scenario," *arXiv preprint arXiv:2103.16223*.

[8] Q. Guo, L. Li, and X. (Jeff) Ban, "Urban traffic signal control with connected and automated vehicles: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 313–334, Apr. 2019.

[9] X. Li, P. Ye, J. Jin *et al.*, "Data augmented deep behavioral cloning for urban traffic control operations under a parallel learning framework," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2021.

[10] A. Kumar, A. Zhou, G. Tucker *et al.*, "Conservative Q-learning for offline reinforcement learning," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1179–1191.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, second edition ed. The MIT Press, 2018.

[12] P. Varaiya, "Max pressure control of a network of signalized intersections," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 177–195, Nov. 2013.

[13] H. Wei, C. Chen, G. Zheng *et al.*, "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 - KDD '19*, 2019, pp. 1290–1298.

[14] L. Li, Y. Lin, N. Zheng *et al.*, "Parallel learning: A perspective and a framework," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 389–395, 2017.

[15] A. Kumar, J. Fu, G. Tucker *et al.*, "Stabilizing off-policy Q-learning via bootstrapping error reduction," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, no. 1055, pp. 11 784–11 794.

[16] J. Fu, A. Kumar, M. Soh *et al.*, "Diagnosing bottlenecks in deep Q-learning algorithms," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 2021–2030.

[17] P. A. Lopez, M. Behrisch, L. Bieker-Walz *et al.*, "Microscopic traffic simulation using SUMO," in *Proceedings of the 21st International Conference on Intelligent Transportation Systems*, 2018, pp. 2575–2582.