International Journal of Intelligent Transportation Systems Research
https://doi.org/10.1007/s13177-022-00321-5

Intelligent Traffic Light via Policy-based Deep Reinforcement Learning

Yue Zhu[1] · Mingyu Cai[2] · Chris W. Schwarz[3] · Junchao Li[1] · Shaoping Xiao[4]

Received: 15 January 2022 / Revised: 13 June 2022 / Accepted: 27 July 2022

Abstract

Intelligent traffic lights in smart cities can optimally reduce traffic congestion. In this study, we employ reinforcement learning to train the control agent of a traffic light on a simulator of urban mobility. As a difference from existing works, a policy-based deep reinforcement learning method, Proximal Policy Optimization (PPO), is utilized rather than value-based methods such as Deep Q Network (DQN) and Double DQN (DDQN). First, the obtained optimal policy from PPO is compared to those from DQN and DDQN. It is found that the policy from PPO performs better than the others. Next, instead of fixed-interval traffic light phases, we adopt light phases with variable time intervals, which result in a better policy to pass the traffic flow. Then, the effects of environment and action disturbances are studied to demonstrate that the learning-based controller is robust. Finally, we consider unbalanced traffic flows and find that an intelligent traffic light can perform moderately well for the unbalanced traffic scenarios, although it learns the optimal policy from the balanced traffic scenarios only.

Keywords Traffic light control · Reinforcement learning · Policy

1 Introduction

As the foundation of our society, transportation systems help ensure that people can reach every destination. Furthermore, transportation promotes economic growth via increasing business productivity, enhancing accessibility of the labor force and jobs, and improving supply chain efficiency. However, traffic congestion has become more and more costly. According to data analyzed by INRIX [1], traffic congestion has cost each American 97 h and $1,348 per year. In addition to the waste of fuel, traffic congestion increases carbon emissions [2, 3], one of the most harmful effects on the environment. Since traffic lights have been used to control traffic flow, one way to mitigate traffic congestion is by maximizing the traffic light performance with an optimal control strategy.

The earliest traditional traffic light control approach includes predefined fixed-time plans [4] in which formulas were derived to predict the average delay to vehicles with the consideration of fixed-cycle traffic lights. In another approach, Cools et al. [5] implemented self-organizing traffic lights via actuated control in an advanced traffic simulator with real data. On the other hand, Zhou et al. [6] investigated adaptive traffic light control of multiple intersections using real-time traffic data. The results demonstrated that the adaptive control could produce lower average waiting time and fewer stops than the predefined fixed-time control and the actuated control. In addition, another adaptive traffic light control [7] was proposed for connected and automated vehicles at isolated traffic intersections. This control approach not only reduces the average waiting time but also guarantees the worst-case waiting time. Furthermore, Demitrov

✉ Shaoping Xiao
shaoping-xiao@uiowa.edu

Yue Zhu
yue-zhu@uiowa.edu

Mingyu Cai
mic221@lehigh.edu

Chris W. Schwarz
chris-schwarz@uiowa.edu

Junchao Li
junchao-li@uiowa.edu

[1] Department of Mechanical Engineering, The University of Iowa, 3131 Seamans Center, Iowa City, IA 52242, USA

[2] Department of Mechanical Engineering, Lehigh University, 113 Research Drive, Bethlehem, PA 18015, USA

[3] National Advanced Driving Simulator, The University of Iowa, 2401 Oakdale Blvd, Coralville, IA 52241, USA

[4] Department of Mechanical Engineering, Iowa Technology Institute, The University of Iowa, 3131 Seamans Center, Iowa City, IA 52242, USA

Published online: 12 August 2022

[8] developed a method to improve the level of traffic light service by optimizing the phase length and cycle.

Riding the wave of artificial intelligence (AI), deep learning (DL) and reinforcement learning (RL) have been employed in solving various engineering problems [9, 10]. As a subset of machine learning, RL [11] enhances the control agent to obtain an optimal action strategy, i.e., policy, during the interaction with the environment. There is a growth of interest in learning-based control of traffic lights [12, 13], i.e., intelligent traffic lights. Li et al. [14] proposed algorithms to design traffic signal timing plans by setting up a deep neural network (DNN) to learn the state-action value function (also called Q-function) of RL. In this deep Q-learning approach, the agent learned appropriate signal timing policies from the sampled traffic state, control inputs, and the corresponding traffic system performance output. Wei et al. [15] also employed deep Q-learning to train the traffic light control agent. They tested the method on a large-scale real traffic dataset obtained from surveillance cameras.

In addition, multi-agent reinforcement learning (MARL) was employed to coordinate the traffic light controllers of multiple intersections. Wu et al. [16] proposed a novel algorithm for traffic light control in vehicular networks. They considered both the vehicles and the pedestrians who waited to pass through the intersection. The experimental results showed that their method could run stably in various scenarios. Wang et al. [17] designed a graph neural network-based model to represent interactions between multiple traffic lights. Then, a deep Q-learning method was utilized to make operation decisions for each traffic light. Chen et al. [18] used a concept of "pressure" in RL so that the designed control agents could coordinate multiple traffic lights. They experimented on a real-world scenario with more than two thousand traffic lights in Manhattan, New York City.

In this study, we utilize a deep RL method to synthesize an optimal operation strategy of a traffic light to pass the traffic flow intelligently. The contributions are manifold, as described below. Most existing works employed value-based RL methods, such as deep Q-learning [15, 17, 18], to train the control agent. As a difference, we adopt a policy-based RL method, Proximal Policy Optimization (PPO), in this work and compare the results with the ones obtained from value-based RL methods. On the other hand, some previous works considered only two traffic light phases [14, 15] or four phases [16] at the intersection. Although only one traffic intersection is studied in this work, we investigate a complex traffic system, which allows left turns, right turns, and U-turns in each branch. Therefore, a total of eight traffic light phases are considered. Furthermore, we consider a traffic light with variable-interval phases, and it performs better than those with fixed-interval phases that have been used in most previous studies [15, 19], mainly resulting in fewer stops. In addition, two traffic scenarios are studied to demonstrate the robustness of the intelligent traffic light trained via PPO: (1) environment disturbance because of car accidents and (2) action disturbance due to traffic light malfunction. According to our best knowledge, neither has been investigated in the literature. Finally, we find that an intelligent traffic light can operate moderately well under unbalanced traffic flows, although it only learns from the balanced traffic scenarios.

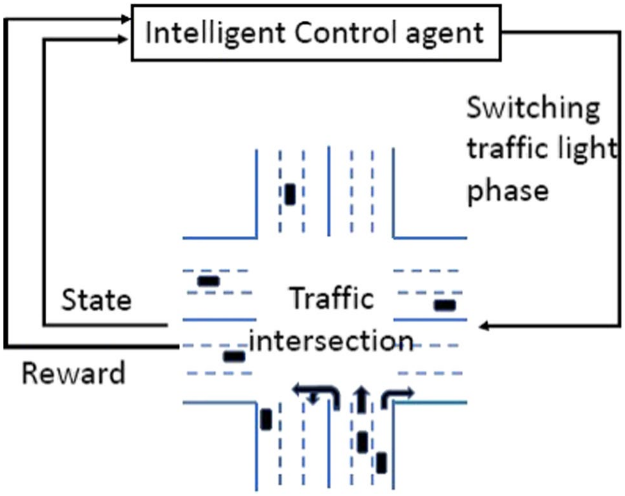

Fig. 1 The interaction between the control agent and the traffic intersection in reinforcement learning

The paper is organized as follows. Section 2 introduces deep RL methods, including value-based and policy-based methods, for the control of traffic lights. Section 3 compares the optimal policies obtained from various RL methods. Then, the advantages of the traffic light phases with variable time intervals in passing the traffic flow are investigated. Section 3 also considers environment and action disturbances to demonstrate the robustness of the optimal policies. Finally, Section 4 provides conclusions and future works.

2 Methodology

2.1 Reinforcement Learning Problem of Traffic Light Control

When formulating an RL problem in this study, the control of a traffic light can be described by the traffic light as a controller or a control agent interacting with the environment, i.e., a traffic intersection, as shown in Fig. 1. The learning process is iterative. At each iteration, the control agent observes the state of the traffic intersection and decides on an action—switching to a new traffic light phase or staying with the current phase. The traffic flow at the intersection will be correspondingly changed due to the decision-making

Table 1 Vehicle data in Simulation of Urban Mobility (SUMO)

Length	Min gap	Max acceleration	Max deceleration
3 m	2 m	1 m/s^2	4.5 m/s^2

Table 2 Traffic flow rates (vehicles per hour)

Right turn	Through movement	Left turn	U-turn
480	600	240	120

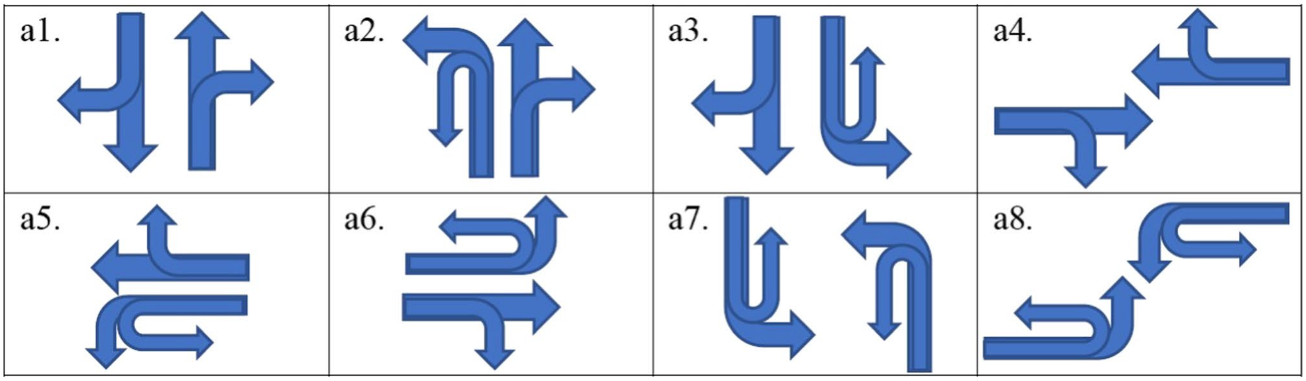

Fig. 2 Traffic light phases

of the control agent. When observing the next state of the traffic intersection, the control agent receives feedback, called a reward, and then decides on the following action.

In addition to an agent, key components in basic RL include an environment, states, actions, and a reward function. In this study, we consider a four-way traffic intersection where there are three 300-m-long lanes in each incoming or outgoing road. The outside lane is for right turns only, while the middle lane is for going straight, i.e., through movement. The inside lane allows both left turns and U-turns, as shown in Fig. 1. This traffic intersection is modeled and simulated via an open-source and highly portable traffic simulator called Simulation of Urban Mobility (SUMO) [20]. It is assumed that all vehicles are the same type with the information in Table 1. The car length is defined as the distance from the front bumper to the rear bumper, while the min gap means the distance between the front car's rear bumper and the rear car's front bumper when stopping. SUMO randomly inputs the vehicles at the end of each incoming road, following the traffic flow setting listed in Table 2, until reaching the desired number of vehicles, which is 808 in this study.

It is assumed that the environment is fully observable, so the control agent has complete knowledge of the traffic status when observing the intersection. Such a so-called state is represented by vehicles' positions, velocities, waiting times, and the current traffic light phase. All information can be withdrawn from the SUMO simulator. Specifically, each road is discretized into many cells [21], and every cell has the same width as the lane width and the length as the summation of the car length and a min gap. If a car's center is located in a cell, 1 is assigned to this cell. Otherwise, 0 is assigned to the cell. Consequently, all cells form a vehicle position matrix in which the number of rows equals the number of roads, and the number of columns is the number of cells on each road. Similarly, the vehicle velocity and waiting-time matrices can be generated by assigning the vehicles' velocities and waiting times in the corresponding cells. In addition, the traffic light phase is encoded as a vector of 1's (green lights) and 0's (red lights). In summary, the state variables include the vehicle position, velocity, and waiting-time matrices, as well as the traffic light phase vector.

The agent can take eight actions to switch the current traffic light phase to one of the phases in Fig. 2. We first consider the traffic light phases with fixed time intervals. There is a 5-s transition if the agent chooses a traffic light phase different from the current one, and the chosen light phase will last 10 s. Otherwise, the current light phase will be extended five more seconds. It shall be noted that later in this study, we will consider the traffic light phases with variable time intervals.

The agent's objective is to find an optimal policy, π^*, which can maximize the expected return for any state as

$$\pi^* = argmax_\pi \ U^\pi \ (s) \tag{1}$$

where $U^\pi(s)$ is the expected discounted return, i.e., the accumulative reward, under policy π starting from state s over the long run and can be calculated as

$$U^\pi(s) = \mathbb{E}^\pi\left[\sum_{i=0}^{\infty}\gamma^i \bullet R\left(s_i, a_i, s_{i+1}\right)\middle| s_0 = s\right] \tag{2}$$

where $\gamma \in [0, 1]$ is a discount factor, and $R(s, a, s')$ is a reward function. The reward function defines the goal of an RL problem, and it provides feedback to the agent about good or bad events after the agent takes action a at state s and reaches next state $s^{'}$. It shall be noted that a policy $\pi = \pi(a|s)$ maps states to actions and provides a guideline for the agent to select actions. Since we consider a fully observable environment with the assumption of Markov property [11], action selection depends on the current state only, and the policy is memoryless.

In this study, the reward function is related to traffic light phase switching and traffic status at the interaction, as expressed below.

$$R = R_a - R_1 - 0.5R_2 + 0.8R_3 \tag{3}$$

where R_a is the action reward. If the agent chooses a different light phase, $R_a = 5$; otherwise, $R_a = 0$. After the agent reaches the next state, the other reward components in Eq. (3) can be calculated based on the vehicle position, velocity, and waiting-time matrices. R_1 is the total number of vehicles that are stopped on all four roads at the next state. A vehicle is considered as stopped if its velocity is below 0.1 m/s. R_2 is the average waiting time (in seconds) of all stopped vehicles by the time of the next state. Once a vehicle starts to move, its waiting time is reset to zero.

The last reward component, R_3, in Eq. (3) is calculated as

$$R_3 = \sum_{i=0}^{N_l} 0.02\left(n_{avg} - n_i\right)n_i \tag{4}$$

where N_l is the total number of lanes at the traffic intersection, $n_{avg} = R_1/N_l$ is the average number of stopped vehicles, and n_i is the number of stopped vehicles at lane i. It shall be noted that R_a, R_1, and R_2 are commonly used in existing works [22]. The other widely used reward components in studies of intelligent traffic lights include the total length of waiting vehicles and the number of vehicles that have passed the traffic light [15]. R_3 is newly introduced in this study. Indeed, R_1 globally quantifies the traffic congestion at the intersection while R_3 represents how well the local traffic congestion is balanced in individual lanes. To the authors' knowledge, this paper is the first to introduce the balance of traffic congestion in the reward of training an intelligent traffic light.

2.2 Q-learning and Value-Based Methods

By maximizing the accumulated reward, the control agent will find the optimal policy to reduce the traffic congestion as much as possible. Using value-based RL methods, we can directly solve so-called optimal value functions (either state value or state-action value) rather than the optimal policies. Indeed, the expected return defined in Eq. (2) is the state value at state s. On the other hand, a state-action value, also called action value or Q value as $Q(s, a)$, is the total reward that an agent can expect to accumulate over a long run, starting from state s and taking action a. Once the optimal value function is found, the optimal policy can be determined via the greedy action selection.

Q-learning [23] is one of the value-based RL methods, and it is model-free because the transition function of state-to-state is not required. This method evaluates all actions at each state to determine the best move via Monte Carlo simulations. The naïve Q-learning method is a tabular method in which Q values at each state are stored in a so-called Q-table. This table can guide the agent to the best action with the highest Q value at each state. In each episode, the Q value at the current state s when taking action a is updated at every step as below based on the Bellman equation [11].

$$\begin{aligned} Q_{new}(s, a) = Q(s, a) + \alpha[R\left(s, a, s^{'}\right) \\ +\gamma max_{a^{'}} Q\left(s^{'}, a^{'}\right) - Q(s, a)] \end{aligned} \tag{5}$$

where $max_{a'}\, Q(s', a')$ outputs the highest Q value at the next state s', γ is the discount factor as defined in Eq. (2), and α is the learning rate.

A discount factor is a number between 0 and 1 so that the total reward remains bounded. It also implies how important future rewards are. In addition, a large learning rate α may have the Q values converge faster. However, the convergence sometimes may be unstable or reach a value function other than the optimal one. On the contrary, a small learning rate can make the converge procedure smoother and more stable but more slowly. In practice, a large learning rate is used at the beginning and then decreases with iterations; this is called an adaptive learning rate.

Given enough episodes in Q-learning, when the optimal state-action value function $Q^{*}(s, a)$ is converged, the optimal policy $\pi^{*}(a|s)$ can be determined by

$$\pi^{*}(a|s) = argmax_a Q^{*}(s, a) \tag{6}$$

Deep neural networks are always employed in RL, named deep reinforcement learning (DRL), to solve the problems in which the state space and/or the action space are enormously large, or infinite (i.e., continuous), such as the traffic state space in this study. Therefore, most existing works [14, 22, 25] in learning-based traffic light control employed value-based DRL methods, including Deep Q Network (DQN) [24] and Double Deep Q Network (DDQN) [25], which are extensions of Q-learning. Since the state space is continuous, it is impossible to utilize the tabular approach to store and withdraw Q values. Unlike the naïve Q-learning that uses a Q-table, DQN and DDQN employ artificial neural networks, called Q-networks, to map states to Q values.

DQN has two Q-networks, an evaluation Q-network and a target Q-network, which have the same architecture, as shown in Fig. 3. The input features, including vehicle position, velocity, and waiting-time matrices, are processed via a

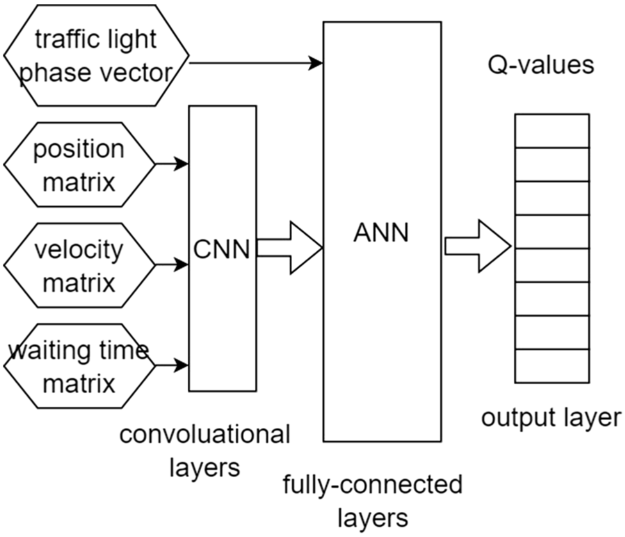

Fig. 3 The architecture of Q-networks

convolutional neural network [26] with two layers. The first layer uses a kernel size of 4 by 4, a stride length of 2, and 16 out channels. The second layer uses a kernel size of 2 by 2, a stride length of 1, and 32 out channels. The rectified linear unit (ReLU) [27] is used as the activation function. The output matrix and the traffic light phase vector are then flattened and passed to a fully-connected neural network to predict Q values. The fully-connected neural network has four hidden layers with 512, 256, 128, and 64 neurons. Since there are eight traffic light phases the control agent can choose, the output layer has eight neurons to predict Q values for individual actions. The ReLU activation function is also used in the fully-connected neural network except for the output layer, in which there is a linear activation function.

During the learning process, the evaluation Q-network is updated at each step by randomly selecting a batch of data samples. At the same time, the target Q-network holds fixed weights until copying from the evaluation Q-network once in a while (every 50 steps in this study). At each step, the evaluation Q-network predicts Q values at the current state s so that the agent can choose an action a at this state via the ε-greedy technique [11]. After reaching the next state s', the Q-networks are used to calculate the Q value when taking action a at state s via a revision of Eq. (5) as

$$Q_{new}(s,a) = Q_e(s,a) + \alpha[R(s,a,s') \\ +\gamma max_{a'} Q_t(s',a') - Q_e(s,a)] \tag{7}$$

where Q_e represents the Q values predicted from the evaluation Q-network, and Q_t represents the Q values predicted from the target Q-network. Equation (7) generates one data sample, i.e., one experience, at each step. As an off-policy RL method, DQN also adopts experience replay memory [28], which makes learning efficient. When applying DQN in this study, a batch of 32 experiences is randomly selected from the experience memory at each step to update the evaluation Q-network.

However, DQN sometimes overestimates Q values. Thus, DDQN [25] is proposed by modifying how to update Q values. Recall that DQN directly uses the maximum Q-value at the next state s' from the target Q-network on the right-hand side of Eq. (7) to update the Q value of action a at the current state s. As a difference, shown in Eq. (8), DDQN selects the action with the highest Q value at the next state s' from the evaluation Q-network and then uses the Q value of the selected action at state s' from the target Q-network to update the Q value of action a at state s.

$$Q_{new}(s,a) = Q_e(s,a) + \alpha[R(s,a,s') \\ +\gamma Q_t(s', argmax_{a'} Q_e(s',a')) - Q_e(s,a)] \tag{8}$$

2.3 Policy-Based Methods

Unlike the value-based RL method, policy-based RL methods directly update and converge the optimal policy. Usually, they have good convergence properties and can learn stochastic policies, defined as $\pi_\theta(a|s)$ with a vector of policy parameters θ, representing the probability of action a to be chosen at state s. A commonly used loss function in policy gradient methods [29] is empirically averaged over a finite batch of experiences

$$L(\theta) = \mathbb{E}_t\left[\log\pi_\theta(a_t|s_t)\widehat{A}_t\right] \tag{9}$$

where $\widehat{A}_t$ is an estimator of the advantage function at time t. Differentiating the objective function, i.e., the loss function, in Eq. (9) results in a gradient estimator [30, 31] during the optimization procedure to update policy parameters. However, the advantage function estimate $\widehat{A}_t$ is usually very noisy, leading to destructively large policy updates.

Schulman et al. [32] proposed a trust region policy optimization (TRPO), in which a surrogate objective is maximized subject to a constraint on the limit of policy updates. Such an optimization problem is expressed as

$$max_\theta \mathbb{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\widehat{A}_t\right] \\ s.t.\mathbb{E}_t\left[KL\left(\pi_{\theta_{old}}(\bullet|s_t), \pi_\theta(\bullet|s_t)\right)\right] \le \delta \tag{10}$$

where θ_{old} is the vector of policy parameters before the update, and KL represents Kullback–Leibler divergence to measure the relative difference between the current and the old policies at a given state s_t.

While keeping the benefits of TRPO, a new family of policy gradient methods, called Proximal Policy Optimization (PPO) [33], revise the objective function in Eq. (10)

as unconstrained optimization problems so that they are easy to implement and have a good sample complexity. One approach uses a penalty on the KL divergence with the adaptive penalty coefficient to achieve a target value of the KL divergence. In this study, we adopt another approach, developing a clipped surrogate objective, which performs better than the objective function with KL penalty.

Let $r_t(\theta) = \pi_\theta / \pi_{\theta_{old}}$ denote the probability ratio of the current policy and the old policy. The clipped surrogate objective in PPO can be written as

$$L^C(\theta) = \mathbb{E}_t[\min(r_t(\theta)\widehat{A}_t, clip\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\widehat{A}_t)] \tag{11}$$

where $clip\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)$ removes the incentive for moving r_t outside of the interval $[1-\epsilon, 1+\epsilon]$, i.e., clipping the probability ratio to modify the surrogate objective. Such a way results in a lower bound on the original surrogate objective and guarantees the objective improvement. We use ϵ=0.15 as the clipping range in this study.

This method is an on-policy learning method and uses every collected experience only once. Consequently, deep neural networks are updated once enough data samples are collected, and all old experiences will be discarded after updating. The network updating procedure can be described as follows. At each step, the current state is the input into the actor-new network to predict the probabilities of actions with which the agent can select the move. After taking the selected action, the agent reaches the next state and receives a reward. The current state, action, and reward are stored as one experience. This process is iterated for a certain number of steps until enough experiences have been collected. This study uses a batch size of 100 to train and update the actor-new and critic networks. The actor-old network is a copy of the actor-new network before updating, and it represents the old policy in the probability ratio function $r_t(\theta)$. During the updating of the actor-new network, the actor-old network always remains the same.

3 Simulations and Discussions

In this study, Monte Carlo simulations with up to 600 episodes are conducted to train the control agent of traffic lights via RL. Each episode terminates when all vehicles pass the traffic interaction. All simulations are performed on a High-Performance Computing (HPC) cluster that is equipped with a GPU of 2080ti and a CPU with a frequency of 2.1 GHz. We first compare the performances of control agents trained via three RL methods–DQN, DDQN, and PPO–to show that PPO is better than the other two methods. Then, the traffic light phases with variable time intervals are implemented, and the induced optimal policy is assessed. Furthermore, environment and action disturbances are studied to investigate the robustness of learning-based traffic lights. The above studies mainly consider balanced traffic flows that correspond to the SUMO setting described in Table 2. Finally, we investigate the scenarios of unbalanced traffic flows. Some simulation videos are provided[1] to demonstrate the operations of intelligent traffic lights under induced optimal policies.

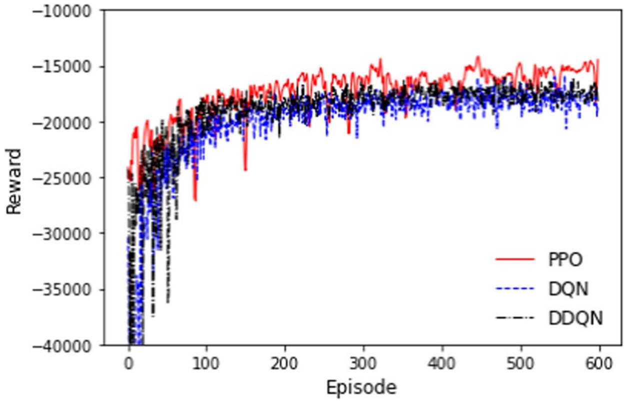

Fig. 4 Reward evolution during the training via DQN, DDQN, and PPO

3.1 Intelligent Traffic Lights Trained via Different DRL Methods

Solving an RL problem aims to achieve an optimal policy, maximizing the accumulated reward over the long run. To compare the training performances of DQN, DDQN, and PPO, the collected reward as a function of episodes is illustrated in Fig. 4. The reward evolution represents the convergence of the training process. Once the training is converged, the accumulated reward can be one of the metrics to assess the performances of various RL methods if the problem setting remains the same. A higher reward means that the induced policy is better. According to Fig. 4, DQN and DDQN have similar rewards after convergence, and PPO results in a higher reward than the others. In other words, PPO is better than DQN and DDQN at training intelligent control agents in this study.

To assess the optimal policies obtained from three different DRL methods, we conduct ten simulations under each policy, respectively, to average the accumulated reward, the time with which all vehicles pass the traffic intersection, and the vehicles' total waiting time. The setting of the SUMO environment remains the same as the

[1] https://github.com/YueZhu95/Intelligent-Traffic-Light-via-Reinforcement-Learning

Table 3 Assessment data of the predefined traffic light and the learning-based traffic lights via DQN, DDQN, and PPO

Method	Reward	Passing time	Waiting time
Predefined	N/A	1,800 s	132,713 s
DQN	-15,215	1,252 s	77,570 s
DDQN	-15,211	1,246 s	76,026 s
PPO	-14,570	1,181 s	59,677 s

one used in learning. Additionally, we also collect the vehicles' passing time and waiting time from a predefined traffic light as a baseline. The predefined traffic light is set to repeatedly loop the traffic light phase from the first phase to the last one, as defined in Fig. 2. The comparison is shown in Table 3.

It can be seen that the policies induced from DQN and DDQN result in almost the same reward, being consistent with the observation from Fig. 4. Also, when the intelligent traffic light operates under either the DQN- or DDQN-induced policies, it takes about 1,250 s to pass all vehicles, i.e., a 30% time reduction compared to the predefined traffic light. Considering the total of the vehicles' waiting time, the DQN- and DDQN-induced policies result in 41% and 43% less time, respectively, than the predefined traffic light control strategy. In addition, the same conclusion that the PPO-induced policy is better than the others can be withdrawn from Table 3 as the one from Fig. 4. Therefore, under the PPO-induced policy, the intelligent traffic light can be more efficient at reducing traffic congestion than those under the DQN- and DDQN-induced policies. Specifically, the vehicles' passing time is reduced by 34%, and the vehicles' total waiting time is reduced by 55%, compared to the times resulting from the predefined traffic light.

Furthermore, it is worth comparing the training speed of RL by DQN, DDQN, and PPO on the same HPC cluster. The wall-clock times are recorded when finishing 600 episodes for each method. It takes DQN, DDQN, and PPO 6.5, 7.75, and 3.5 h, respectively, to retrieve the optimal policies. PPO is faster than DQN and DDQN because it doesn't update deep neural networks at each step like DQN and DDQN do. Indeed, as an on-policy learning method, PPO updates the actor and critic networks after a certain number of steps once enough experiences have been collected. Afterward, all the old experiences are discarded. In contrast, DQN and DDQN, which are off-policy learning methods, utilize the technique of experience replay to update Q-networks at each step.

In summary, learning-based intelligent traffic lights perform better than traffic lights with a predefined fixed-time plan. After comparing three RL methods, PPO, a policy-based method, is more efficient and effective than DQN and DDQN, which are value-based methods in training intelligent traffic lights.

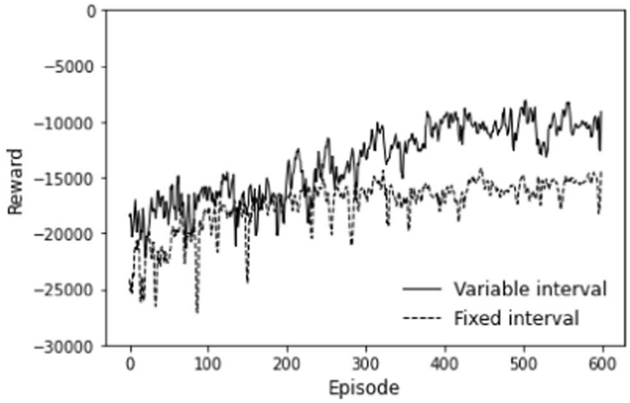

Fig. 5 Reward evolution when considering variable time intervals, compared to the one with fixed time intervals

3.2 Traffic Light Phases with Variable Intervals

The above study only considers the traffic light phases with fixed-time intervals, as in most existing works [22]. Specifically, during the training, a selected traffic light phase stays for a constant interval of 10 s if it is different from the last phase. Although the current traffic light phase has a probability of being chosen and extended for another 5 s, there is not much flexibility in selecting various time intervals for the same traffic light phase.

Here we consider traffic light phases with variable time intervals, including 10, 15, 20, and 25 s for a chosen phase. Consequently, the agent needs to choose both a traffic light phase and an interval for the phase to stay during the action selection at each step. Therefore, instead of 8 available actions, as illustrated in Fig. 2, there are 32 available actions in this study, i.e., 32 combinations of traffic light phases and intervals.

Only the PPO method is used in this study because it has been shown to perform better than DQN and DDQN, and it needs less wall-clock time to converge. The actor and critic neural networks keep the same architectures as in Fig. 3, except that the output layer of the actor-network has 32 neurons instead of 8. Other parameters in training traffic lights via RL are the same as provided above. Figure 5 illustrates the reward evolution during the training via PPO when considering variable time intervals, compared to the one with fixed time intervals. It can be seen that considering traffic light phases with variable intervals results in a higher reward once converged. That means a better policy is achieved.

To quantitatively assess the induced policy when considering the traffic light phases with variable intervals, we conduct ten simulations under the obtained optimal policy and average the time to pass all vehicles and the vehicles' waiting time. The vehicles' passing time and total waiting time are 1,076 s and 55,028 s, respectively. Compared to

Table 4 Assessment data when collisions occur as environment disturbances

Methods	Collision occurrence	Passing time	Waiting time
Predefined	Incoming road	1,820 s	135,117 s
Policy A	Incoming road	1,271 s	69,884 s
Policy B	Incoming road	1,260 s	70,233 s
Predefined	Outgoing road	1,800 s	132,750 s
Policy A	Outgoing road	1,190 s	64,965 s
Policy B	Outgoing road	1,192 s	66,979 s

the data resulting from the traffic light phases with fixed time intervals (the control agent was trained via PPO in Section 2.1, shown in Table 3, the time reductions are 9% in the passing time and 8% in the total waiting time. We also calculate the average number of times the traffic light phases have been switched, i.e., the number of traffic light stops, under each optimal policy. We find that the number is dramatically reduced from 137 to 63 when considering variable time intervals. Although there are no significant improvements in the passing time and waiting time, a much smaller number of action choosing results in a notably higher reward, as shown in Fig. 5.

3.3 Environment and Action Disturbances

This study considers environment and action disturbances to demonstrate that learning-based intelligent traffic lights are robust. Two policies, named A and B, from previous PPO training are adopted. The difference is that Policy A is for the traffic light phases with variable time intervals, while policy B is for the light phases with fixed time intervals.

The environment disturbance may be caused by a traffic collision/car crash or car breakdown. We assume each vehicle has a 0.2% probability of crash or breakdown in this study. Random number generations can introduce such an environment uncertainty in the SUMO simulator. During the simulation, the car crash or breakdown can occur randomly in any lane of either incoming or outgoing roads at an unexpected time. Once a car crash or breakdown occurs, the involved vehicle(s) will stay at the current position(s) for 300 s. At the same time, the vehicles behind them must change lanes to proceed. Consequently, traffic congestion increases because of such an environment disturbance. In addition, it is assumed that the vehicles involved in the crash or breakdown will be removed after 300 s and have no further impact on the traffic flow.

As utilized above, ten simulations with collisions occurring on the incoming roads or the outgoing road are conducted to assess policies A and B, respectively. The results are compared to those from predefined traffic lights, as listed in Table 4. We separate the collision occurrence on the incoming and outgoing roads into two scenarios because collisions on the incoming roads have a higher impact on the traffic flow than collisions on the outgoing roads.

Table 4 shows that learning-based traffic lights perform much better than predefined traffic lights, just as concluded from the above studies. Traffic lights under Policies A and B perform similarly with respect to the vehicles' passing time. In addition, Policy A is slightly better than Policy B at reducing the vehicles' total waiting time. It should be noted that the number of traffic light stops under Policy A is much smaller than the one under Policy B because variable-interval light phases are utilized in Policy A. An interesting phenomenon we observe in this study is that we retrain the traffic light by implementing the environment disturbances in training, but the new policy performs worse than Policies A and B, although it is better than the predefined traffic light control strategy.

Action disturbances, i.e., action uncertainties, can occur due to the malfunction of the traffic light. In this study, after the control agent selects a light phase to switch or stay, the traffic light has a 90% probability of switching to the desired light phase and another 10% probability of randomly switching to one of the other phases. Consequently, the optimal operation policy cannot be exactly followed. Introducing such action disturbance will also deteriorate traffic congestion. It shall be noted that the environment is fully observable and the control agent can observe the actual traffic light status. Therefore, the current traffic light phase vector, one of the state variables to determine the next action, is based on the actual traffic light phase instead of the one previously determined by the agent.

In addition to Policies A and B, a new policy (Policy C) is obtained via PPO, implementing an action disturbance in the SUMO simulator for training. Unlike Policies A and B, which are learned from a perfect traffic environment without action disturbances, Policy C is learned from the traffic environment considering a probability of traffic light malfunction (i.e., a 10% probability in this study). In addition, Policies A and C are for traffic lights with variable-time-interval phases, while Policy B is for the light with fixed-time-interval phases. The performances of all three policies are evaluated in the traffic environment in which the action disturbances are implemented. The assessment data of the three policies, compared to the predefined traffic light, are listed in Table 5. Again, the learning-based traffic lights under Policies A, B, and C perform much better than the predefined traffic light, considering action disturbances due to traffic light malfunctions. Policy A is slightly better than Policy B; however, Policy A results in fewer light stops than Policy B. Although it considers action disturbance in training, the induced policy (Policy C) doesn't operate the traffic light better than Policies A and B. However, Policy C

Table 5 Assessment data when action disturbances occur due to traffic light malfunction

Methods	Passing time	Waiting time	# of phase switches
Predefined	1,942 s	139,863 s	129
Policy A	1,187 s	58,389 s	65
Policy B	1,240 s	60,932 s	132
Policy C	1,348 s	76,640 s	130

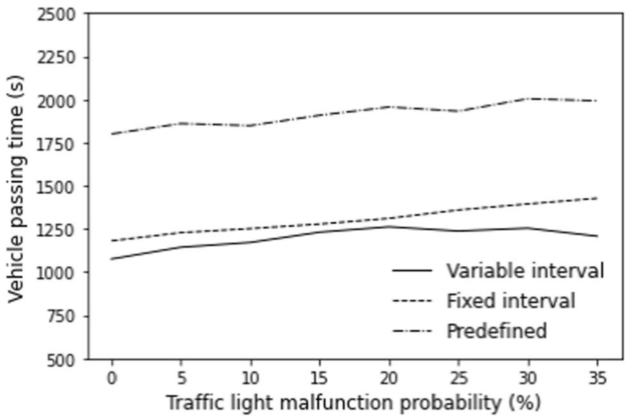

Fig. 6 Vehicle passing times considering various probabilities of traffic light malfunction

may be useful in practice because it is from online training, while Policies A and B are induced from offline training.

In addition, we investigate the traffic light performances under the predefined plan, Policy A, and Policy B under various probabilities of action disturbances, i.e., considering different traffic light malfunction probabilities, from 5 to 35% at 5% intervals. We conduct 50 simulations for every policy at each light malfunction probability and plot the averaged vehicle passing times in Fig. 6. It can be seen that Policy A results in similar passing times with a traffic light malfunction probability up to 35%, while the vehicle passing times under Policy B and the predefined plan gradually increase as the light malfunction gets worse. However, traffic lights under Policies A and B perform much better than the predefined traffic light, as concluded above. It should be noted that based on our observation, when the light malfunction probability becomes larger than 35%, the vehicle passing time under Policy A is notably increased as well.

3.4 Unbalanced Traffic Flow

In the above studies, only balanced traffic flows are considered. In other words, each incoming road has the same traffic flow rates as indicated in Table 2. In addition, Policy A was learned from such a perfect traffic environment with balanced traffic flows. This study considers a complex traffic environment that includes balanced and unbalanced traffic flows. A new policy, Policy D, is learned from this complex traffic environment, in which the SUMO simulator inputs vehicles for 1,500 s in three stages. During the first 500 s, the same balanced flow rates as in Table 2 are adopted. Then, the flow rates on the incoming east road are reduced by three-fourths for another 500 s while the flow rates remain the same on other incoming roads. Finally, during the third 500 s, the flow rates on the incoming east road are back to normal, while the other incoming roads' flow rates are reduced by three-fourths.

Table 6 Assessment data of traffic lights under the predefined plan, Policy A, and Policy D in complex traffic flows consisting of balanced and unbalanced traffic

Methods	Passing time	Waiting time
Predefined	4,320 s	452,401 s
Policy A	3,041 s	233,483 s
Policy D	2,790 s	211,528 s

To evaluate the performances of Policies A and D, both are applied to the same complex traffic environment as described above. For comparison, we also utilize the predefined traffic light indicated in Section 3.1. The averaged vehicle passing times and waiting times are compared in Table 6. Although Policy A is obtained from the learning with balanced traffic flows, the traffic lights under this policy perform moderately well, compared to those under Policy D, which is particularly learned from complex traffic flows consisting of balanced and unbalanced traffic. Specifically, the vehicle passing time and waiting time from Policy D are only 8.3% and 9.4%, respectively, which are better than Policy A's. It is acceptable that policies learned from balanced traffic flows can also handle unbalanced traffic flows. However, when considering extremely unbalanced traffic flows, our other simulations[2] indicate that it would be better to learn the optimal policy directly from these scenarios.

4 Conclusions and Future Works

In this research, an intelligent traffic light learns to operate properly at a traffic intersection via RL. After comparing the performance of three DRL methods, DQN, DDQN, and PPO, and assessing their induced policies, PPO as a policy-based DRL method is better than value-based DRL methods such as DQN and DDQN. We consider various time intervals

[2] github.com/YueZhu95/Intelligent-Traffic-Light-via-Reinforcement-Learning.

from which the control agent can choose for any light phase. Compared to the fixed-interval traffic lights that most existing works assumed, the traffic light with variable-interval phases generally can result in shorter passing times, shorter vehicle waiting times, and a much smaller number of phase switches. We also study the scenarios in which there are environment disturbances due to collisions or action disturbances because of traffic light malfunction. Our simulations demonstrated that the optimal policies via offline training without disturbances were robust and performed well in those scenarios.

This paper focuses on learning-based traffic light control at a single traffic intersection. Multiple traffic lights will be studied in the future, and multi-agent reinforcement learning (MARL) needs to be adopted. It should be noted that a limited number of intervals are considered in this study so that the action space is still discrete. This work can be extended to constructing a continuous action space in which intervals within a time range are available. Consequently, some other policy-based methods, including Asynchronous Advantage Actor Critic (A3C) and Deep Deterministic Policy Gradient (DDPG) could be applied. In addition, both vehicles and pedestrians could be considered in a future work to design learning-based traffic lights.

Acknowledgements Xiao thanks the supports from the US Department of Education (ED#P116S210005).

Authors' contributions YZ initiated the study, carried out methodology development and simulations, and drafted the initial manuscript. MC conceived of the study and helped to draft the manuscript. CWS conceived of the study and helped to draft the manuscript. JL helped to draft the manuscript. SX supervised the study and was a major contributor in finalizing the manuscript. All authors read and approved the final manuscript.

Funding The US Department of Education (ED#P116S210005).

Data availability The simulation videos can be found on the website link provided in the manuscript. Some or all data, models, or codes that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Ethics Approval and Consent to Participate Not applicable.

Consent for Publication All authors have approved the manuscript and gave their consent for submission and publication.

Competing Interests The authors declare no competing financial interests.

References

1. INRIX: Congestion Costs Each American 97 hours, $1,348 A Year - INRIX. https://inrix.com/press-releases/scorecard-2018-us/. Accessed October 5, 2021.
2. Zhang, K., Batterman, S.: Air pollution and health risks due to vehicle traffic. Sci Total Environ. **450–451**, 307–316 (2013). https://doi.org/10.1016/J.SCITOTENV.2013.01.074
3. Bharadwaj, S., Ballare, S., Rohit, Chandel, M.K.: Impact of congestion on greenhouse gas emissions for road transport in Mumbai metropolitan region. Transp Res Procedia. **25**, 3538–3551 (2017). https://doi.org/10.1016/J.TRPRO.2017.05.282
4. Miller, A.J.: Settings for Fixed-Cycle Traffic Signals. J Oper Res Soc. **14**(4), 386 (1963). https://doi.org/10.2307/3006800
5. Cools, S.B., Gershenson, C., D'Hooghe, B.: Self-Organizing Traffic Lights: A Realistic Simulation. In: Prokopenko M, ed. Advanced Information and Knowledge Processing. Springer, London; 45–55. (2013). https://doi.org/10.1007/978-1-4471-5113-5_3
6. Zhou, B., Cao, J., Wu, H.: Adaptive traffic light control of multiple intersections in WSN-based ITS. IEEE Veh Technol Conf. (2011). https://doi.org/10.1109/VETECS.2011.5956434
7. Miao, L., Leitner, D.: Adaptive Traffic Light Control with Quality-of-Service Provisioning for Connected and Automated Vehicles at Isolated Intersections. IEEE Access. **9**, 39897–39909 (2021). https://doi.org/10.1109/ACCESS.2021.3064310
8. Dimitrov, S.: Optimal Control of Traffic Lights in Urban Area. 2020 Int Conf Autom Informatics, ICAI 2020 - Proc. October 2020. https://doi.org/10.1109/ICAI50593.2020.9311318
9. Xiao, S., Hu, R., Li, Z., Attarian, S., Björk, K.-M., Lendasse, A.: A machine-learning-enhanced hierarchical multiscale method for bridging from molecular dynamics to continua. Neural Comput Appl. **32**(18), 14359–14373 (2020). https://doi.org/10.1007/S00521-019-04480-7
10. Cai, M., Hasanbeig, M., Xiao, S., Abate, A., Kan, Z. Modular Deep Reinforcement Learning for Continuous Motion Planning with Temporal Logic. IEEE Robot. Autom. Lett. 6(4):7973–7980. (2021). http://arxiv.org/abs/2102.12855. Accessed April 8, 2021
11. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. The MIT Press, London (2018)
12. Bingham, E.: Reinforcement learning in neurofuzzy traffic signal control. Eur J Oper Res. **131**(2), 232–241 (2001). https://doi.org/10.1016/S0377-2217(00)00123-5
13. Kuyer, L., Whiteson, S., Bakker, B., Vlassis, N.: Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs. Lect Notes Comput Sci. **5211**, 656–671 (2008). https://doi.org/10.1007/978-3-540-87479-9_61
14. Li, L., Lv, Y., Wang, F.Y.: Traffic signal timing via deep reinforcement learning. IEEE/CAA J Autom Sin. **3**(3), 247–254 (2016). https://doi.org/10.1109/JAS.2016.7508798
15. Wei, H., Yao, H., Zheng, G., Li, Z.: IntelliLight: A reinforcement learning approach for intelligent traffic light control. Proc ACM SIGKDD Int Conf Knowl Discov Data Min. 2496–2505. (2018). https://doi.org/10.1145/3219819.3220096
16. Wu, T., Zhou, P., Liu, K., et al.: Multi-Agent Deep Reinforcement Learning for Urban Traffic Light Control in Vehicular Networks. IEEE Trans Veh Technol. **69**(8), 8243–8256 (2020). https://doi.org/10.1109/TVT.2020.2997896
17. Wang, Y., Xu, T., Niu, X., Tan, C., Chen, E., Xiong, H.: STMARL: A Spatio-Temporal Multi-Agent Reinforcement Learning Approach for Cooperative Traffic Light Control. IEEE Trans Mob Comput. 1–1 (2020). https://doi.org/10.1109/TMC.2020.3033782
18. Chen, C., Wei, H., Xu, N., et al.: Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control. Proc AAAI Conf Artif Intell. **34**(04), 3414–3421 (2020). https://doi.org/10.1609/AAAI.V34I04.5744
19. Wei, H., Xu, N., Zhang, H., et al.: Colight: Learning network-level cooperation for traffic signal control. Int Conf Inf Knowl Manag Proc. 1913–1922. (2019). https://doi.org/10.1145/3357384.3357902

20. Lopez, P.A., Behrisch, M., Bieker-Walz, L., et al.: Microscopic Traffic Simulation using SUMO. IEEE Conf Intell Transp Syst Proceedings, ITSC. **2018**, 2575–2582 (2018). https://doi.org/10.1109/ITSC.2018.8569938
21. Liang, X., Du, X., Wang, G., Han, Z.: A Deep Reinforcement Learning Network for Traffic Light Cycle Control. IEEE Trans Veh Technol. **68**(2), 1243–1253 (2019). https://doi.org/10.1109/TVT.2018.2890726
22. Nishi, T., Otaki, K., Hayakawa, K., Yoshimura, T.: Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets. IEEE Conf Intell Transp Syst Proceedings, ITSC. **2018**, 877–883 (2018). https://doi.org/10.1109/ITSC.2018.8569301
23. Watkins, C., Dayan, P.: Q-Learning. Mach Learn. **8**, 279–292 (1992). https://doi.org/10.1007/BF00992698
24. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Playing Atari with Deep Reinforcement Learning. https://arxiv.org/abs/1312.5602v1 (2013). Accessed September 19, 2021
25. Hasselt H van, Guez, A., Silver, D.: Deep Reinforcement Learning with Double Q-Learning. In: *AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. **30**, 2094-2100 (2016)
26. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc IEEE. **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
27. Nair, V., Hinton, G.: Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. **32**, 807–814 (2010)
28. Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Mach Learn. **8**(3), 293–321 (1992). https://doi.org/10.1007/BF00992699
29. Kakade, S., Langford, J.: Approximately optimal approximate reinforcement learning. In: In Proc. 19th International Conference on Machine Learning (2002)
30. Mnih, V., Badia, A.P., Mirza, M., et al.: Asynchronous Methods for Deep Reinforcement Learning. In: Balcan MF, Weinberger KQ, eds. Proceedings of The 33rd International Conference on Machine Learning. Vol 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR:1928–1937. https://proceedings.mlr.press/v48/mniha16.html (2016). Accessed November 23, 2020
31. Schulman, J., Moritz, P., Levine, S., Jordan, M.I., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. In: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings. International Conference on Learning Representations, ICLR. https://sites.google.com/site/gaepapersupp (2016). Accessed November 23, 2020
32. Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P.: Trust Region Policy Optimization. 32nd Int Conf Mach Learn ICML 2015. **3**,1889–1897. http://arxiv.org/abs/1502.05477 (2015). Accessed November 23, 2020
33. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. *arXiv*. https://arxiv.org/abs/1707.06347v2 (2017). Accessed November 23, 2020

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.