Reference RL

Reinforcement learning with reference mechanism and its application in traffic signal control

Lu, Yunxue; Hegyi, Andreas; Maria Salomons, A.; Wang, Hao

DOI
10.1016/j.ins.2024.121485

Publication date
2024

Document Version
Final published version

Published in
Information Sciences

Information Sciences 689 (2025) 121485

Contents lists available at ScienceDirect

Information Sciences

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

Reference RL: Reinforcement learning with reference mechanism and its application in traffic signal control

Yunxue Lu [a,b,c], Andreas Hegyi [d], A. Maria Salomons [d], Hao Wang [a,b,c,*]

[a] Jiangsu Key Laboratory of Urban ITS, Southeast University, 2 Si Pai Lou, Nanjing 210096, PR China
[b] Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, Southeast University, 2 Si Pai Lou, Nanjing 210096, PR China
[c] School of Transportation, Southeast University, 2 Si Pai Lou, Nanjing 210096, PR China
[d] Department of Transport and Planning, Delft University of Technology, P.O. Box 5048, 2600 GA Delft, the Netherlands

ARTICLE INFO

Keywords:
Traffic signal control
Reinforcement learning
Real-world learning
Reference policy
Reference mechanism

ABSTRACT

This paper addresses the challenges of deploying reinforcement learning (RL) models for traffic signal control (TSC) in real-world environments. Real-world training can prevent mismatches between simulation environments and the actual traffic conditions, thereby achieving better performance of agent upon deployment. However, free explorations by agents during real-world training can disrupt traffic operations. To mitigate this, this paper proposes a reference mechanism to guide the decision-making process within the RL framework. A reference timing policy, typically a model-based signal strategy, is integrated into the learning process to provide agents with reference actions. Specifically, an additional Q-value function is introduced to evaluate both the agent's actions and those of the reference policy, allowing for adjustments before the actions are executed in real traffic system. Numerical results indicate that the reference mechanism effectively enhances system performance early in the training process, thus accelerating learning. We also combine the reference RL method with a pretraining procedure and a jump-start algorithm, respectively. Experimental results demonstrate their effectiveness in further enhancing system performance and facilitating real-world training.

1. Introduction

The traffic signal control problem (TSC) has been extensively studied due to its vital role in separating conflicting traffic flows and facilitating the efficient movement of vehicles through intersections. Given its widespread deployment, traffic signal controllers significantly impact traffic safety, efficiency and pollution levels. Solutions to TSC challenges generally fall into three categories: traffic theory-based, simulation-based, and data-driven methods [1]. Conventional traffic theory-based TSC methods typically formulate analytical models to develop signal plans based on simplified representations of traffic dynamics [2]. These approaches often rely on low-resolution data from traffic detectors, such as induction loops, ultrasonic and radar detection systems. While these methods are highly interpretable and trusted by jurisdictions, their effectiveness are significantly constrained by the low-resolution traffic data and the accuracy of the underlying models. In contrast, simulation-based TSC methods provide a more detailed representation of traffic dynamics by optimizing control strategies through feedback from traffic simulators [3].

* Corresponding author.
E-mail address: haowang@seu.edu.cn (H. Wang).

https://doi.org/10.1016/j.ins.2024.121485
Received 21 January 2024; Received in revised form 14 September 2024; Accepted 15 September 2024
Available online 19 September 2024

Reinforcement learning (RL)-based TSC methods fits in the third category. As a branch of machine learning (ML), RL has gained increasing attention across various scientific and engineering fields. Due to its model-free nature and ability to incorporate time into decision making process, RL offers advantages in automatic learning of optimal decisions over time, including the TSC optimizations. A RL-based TSC agent learns to optimize its timing policy by interacting with the traffic system to maximize the long-term return. However, the complexity of traffic system and its partially observable nature make this learning process challenging. Agents need extensive interactions with the environment before learning a satisfying policy, thus limiting the training process to simulated scenarios. Furthermore, the mismatch between simulation environments and real-world conditions can pose new challenges after deployment, as agents may encounter dynamics that were not adequately modeled during training. This highlights the need for real-world training of RL-based agents, including TSC agents, to enhance their practical applicability and performance. Our main contributions are as follows:

(1) We propose a reference RL framework that incorporates a reference policy into the traditional RL framework, providing reference actions for the agent's decision-making. This allows the traffic system to achieve reasonable performance early in the learning process, significantly improving sample efficiency compared to traditional RL methods that starts from scratch.
(2) An additional Q-value function is developed to evaluate both the agent's actions and those of the reference policy, enabling informed adjustments to actions before they are executed.
(3) A pretraining procedure and the jump-start method [4] are integrated to the reference RL framework, respectively. It helps further promote the system performance during the early training process and facilitates the practice of real-world training.
(4) Simulation experiments demonstrate that different reference policies have varying effects on prompting the system performance during the training process. A simple reference policy that allocates green time in proportion to the number of queuing vehicles on active lanes can also substantially promote the system performance and accelerate the training process.
(5) The proposed framework is not limited to TSC problems; it can be applied to other RL-based control problems. In such cases, the reference policy and RL attributes can be customized to meet specific control requirements.

2. Related works and motivations

This section reviews key studies to gain insights into the advancements of RL-based TSC methods. Additionally, we draw inspiration from other RL-based control domains for facilitating real-world training of TSC controllers.

2.1. Related works

Various innovative designs of RL-based signal controller have been proposed to address different challenges in TSC problems. Some focus on improving agent convergence performance [5], while others reduce model dimensionality to lower computational demands [6,7]. In view of the large-scale deployment of traffic signals in road networks, most of the researches designed the signal controller in a distributed way to overcome the scalability issue of models. These models often need to enhance the communication between agents to mitigate partial observability and improve agent convergence performance [8,9]. Additionally, some studies emphasize speeding up the learning process of agents [10,11]. Generalizability is also of high interest to researchers. It evaluates how the TSC model trained in a particular traffic scenario can adapt to a new scenario [12,13]. Besides, a considerable body of research addresses various domain-specific issues, such as pedestrian crossing [14,15], traffic safety [16,17], and priority control [18,19].

Most RL-based TSC methods, with few exceptions [10,11], adopt a cold-start training process, requiring agents to engage in extensive interactions with the simulated environment before achieving satisfactory policies. Limited attention has been given to techniques that promote real-world training of traffic signal controllers [10]. pretrain TSC agents in an offline way using experience data from an existing model-based controller, boosting initial online performance [11]. designed a cyclical dataset for offline training the TSC agent, avoiding online interactions with the real-world traffic system.

In other RL-based control domains, various techniques aim to guide and accelerate the learning process, offering valuable insights for the design of real-world TSC agents. one technique involves providing initial knowledge for agents to achieve improve agent's early performance, thereby speeding up the training process. Prior knowledge can take the form of demonstration data, multi-task data, or other information sources. Resort to these external knowledges, better initialization of value functions [20–22] and of policy network [23,24], a fitness model of environment [25,26], exploration strategies [27], or state representations [28,29] can be obtained before online learning. Therefore, exhaustive interactions from scratch can be avoided and the learning process can be accelerated as well. In this case, a pretraining-adaptation paradigm is commonly adopted to allow agents to learn transferable skills during pretraining, which then facilitate adaptation to new tasks [30]. Learning from demonstrations (LfD) is a specific case of this paradigm, where demonstration data directly relates to the agent's task. Depending on the quality of demonstration data, the model can be finetuned in "adaptation" stage. More detailed reviews of related studies can be found in [31,32].

Another technique focuses on constraining the agent's exploration by modifying undesired actions to accelerate learning process and reduce safety risks. In many systems, unlimited exploration of agents is impractical due to efficiency or security concerns. To improve system performance and safety, agents may seek teacher advices or guidance from trusted models to modify actions or reshape reward signals to change the probability distribution of actions [33]. A "teacher" can be a human expert on the target task or a trusted method. When to provide guidance and what form of advice to provide, are key issues in teacher advising framework. Common advising mechanisms include uncertainty-based advising [34], novelty-based advising [35], early advising [36], fixed-frequency advising [4] and safety-based advising [37].

While these studies have been demonstrated to effectively improve the initial performance, accelerate learning, or ensure the system safety, they present challenges. Pretraining require vast amounts of experience data, and its effectiveness can be quietly constrained by the quality of the experience data. Moreover, [31] points that bootstrap error in offline learning with actor-critic methods can result in obvious performance drop during fine-tuning. In teacher-involved methods, it is challenging to artificially find appropriate timings and forms of advices from the teacher.

2.2. Motivations

The motivations of this study are based on the following considerations.

(1) The necessity of training TSC agent in real-world environment. Most existing RL-based TSC models adopt a cold-start training on traffic simulators. Although simulators, like SUMO [38] and VISSIM [39], can offer virtual environments for agent training, the mismatch between simulated environment and real-world traffic system often raises concerns regarding the effectiveness of the control model upon deployment. Moreover, it is challenging to quantify the performance degradation resulting from this discrepancy before real-world deployment. Real-world training of TSC agents can effectively prevent the performance degradation caused by environmental mismatches.
(2) Ensuring reliability of the real-world training. For safety–critical system like traffic system, agents learning from scratch may generate undesired signal plans that jeopardize system safety and stability, particularly in the early stages of learning. Therefore, efforts towards real-world training of signal controllers should prioritize two key aspects: maintaining acceptable system performance and achieving fast convergence to prevent severe traffic disruptions. The main insight that we leverage for real-world training of TSC agents is to introduce a trusted reference policy to guide the agent's explorations, providing reference actions for the agent's decision-making. This ensures that, even with limited exploration, the traffic system is able to maintain reasonable performance. Meanwhile, proposed reference mechanism helps achieve a rapid improvement of system performance.
(3) Minimizing the complexity of the reference mechanism. In real-world environments, frequent adjustments to the reference mechanism through experimentation are impractical due to the direct impact on system operations. Therefore, the reference mechanism should be simple, using minimal hyperparameters. To this end, the proposed method establishes a Q-value baseline for the agent's decision-making, pruning poor actions with smaller Q values than the action generated by the reference model. The proposed method is referred to as "reference RL" in this study. The reference RL offers several advantages: ① Pretraining is not a prerequisite before engaging in online interaction, eliminating the necessity for an extensive dataset of offline experience from the reference policy. ② No advising mechanism design. All agent actions need to satisfy the Q-value constraint from the reference policy. ③ No artificially modification of reward signals. ④ Flexibility of the reference model: The reference model need not to be "optimal" but can be any trusted TSC model, simplifying its design and implementation.

3. Theoretical background

This section provides brief overview of RL and the Soft Actor-Critic (SAC) algorithm, a state-of-the-art RL method.

3.1. Reinforcement learning

There are two entities in the framework of reinforcement learning: the agent and the environment. The agent acts as the decision maker, improving its strategy (policy) by interacting with the environment. Everything external the agent is regarded as the environment. Communications between the agent and its environment occurs through three channels [40]: observations (O), actions (A), and rewards (R). Rooted in Markov Decision Process (MDP), the learning process of RL can be represented as a tuple $M = \langle S, O, A, P, R, \gamma \rangle$. S is the state space, and observations O provide partial representations of the environment's state. A represents actions taken by the agent, influencing state transition dynamics $P := S \times A \rightarrow S$ directly. Reward R is the feedback the agent receives from environment after executing an action. γ is the discount factor that balances the immediate rewards with long-term gains. At each time step t, the agent observe the state s_t of system, decides the next action a_t based on its current policy π, and receives a reward r_{t+1} from the environment after performing a_t. databased on these interactions, the agent reinforces its behaviors in a positive or negative way, aiming to achieve the most accumulated reward over time.

A few terminologies helpful to formulate the RL framework are as follows: The return G_t is defined as the discounted sum of future rewards starting from time step t:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k \tag{1}$$

where T is the duration of each training episode. For convenience, the value of a specific state $V(s)$ and the value of a specific state-action pair $Q(s, a)$ are defined as the discounted sum of rewards received in following steps respectively.

$$V(s_t) = E[G_t | s_t] = \sum_{a_t \in A} \pi(a_t | s_t) r(s_t, a) + E\left[\sum_{i=t+1}^{T} \gamma^{i-t} r(s_i, a_i) \right] \tag{2}$$

$$Q(s_t, a_t) = E(G_t|(s_t, a_t)) = r(s_t, a_t) + E\left[\sum_{i=t+1}^{T} \gamma^{i-t} r(s_i, a_i)\right] \tag{3}$$

$\pi(a_t|s_t)$ in (2) is the probability of agent choosing action a_t at state s_t.

3.2. The Soft Actor-Critic algorithm

Value methods and policy methods are two main approaches to learn an optimal control policy. Q-learning, a widely used value-based method iterates over Q-values of state-action pairs $Q(s, a)$ until convergence. The Q-values are updated using the Bellman equation [40]:

$$Q(s_t, a_t) \leftarrow (1-\zeta) Q(s_t, a_t) + \zeta\left[r + \gamma \max_{a_{t+1} \in A} \overline{Q}(s_{t+1}, a_{t+1})\right] \tag{4}$$

Where ζ is a factor balancing old and new Q-value estimates. $\overline{Q}$ is the target network of Q, which is a frozen recent model of Q. After deployment, the action for a given state is generated by $a^* = \text{argmax}_a Q_*(s,a)$. The Q-learning algorithm is commonly applied to control problems with discrete action space. However, TSC problems typically involves continuous action space, making policy-based methods more suitable. The Soft Actor-Critic (SAC) method employed in this paper is a leading policy-based approach. It optimizes the expected return while incorporating an entropy term of policy into the objective function to encourage exploration. Then the optimal policy can be formulated as

$$\pi^* = \underset{\pi_\sigma}{\text{argmax}} \sum_t E_{(s_t, a_t) \sim \pi_\sigma}[Q_\omega(s_t, a_t) + \alpha H(\pi_\sigma(.|s_t))] \tag{5}$$

where σ denotes the parameters of policy π, $\pi_\sigma(.|s_t)$ represents the probability distribution of available actions at the visiting state s_t under the policy π_σ, and $H(\pi_\sigma(.|s_t)) = -\log\pi(.|s_t)$ is the entropy of the policy π_σ at the state s_t, and α is a temperature parameter to scale the entropy measure. The additional entropy term promotes the agent learning a more comprehensive policy and speeding up convergence. With the optimal policy π^*, the optimal action at time step t can be obtained by a forwarding process $a_t^* = \text{argmax}_a \pi_{\sigma^*}(a|s_t)$. Algorithm1 shows the SAC algorithm, which involves three learning functions: policy function, Q-value function and a function dynamically adjusting the temperature α. The SAC uses two Q-networks, parameterized by ω_1 and ω_2 to reduce over-estimation bias in Q-value estimation.

Algorithm 1: Soft Actor-Critic

Input: $\sigma, \omega_1, \omega_2$ # Initial parameters
$\overline{\omega_1} \leftarrow \omega_1, \overline{\omega_2} \leftarrow \omega_2$ # Initialize frozen Q-network weights
D$\leftarrow\varnothing$ # Initialize an empty dataset
for each iteration **do**
  for each environment step **do**
    $a_t \sim \pi_\sigma(a_t|s_t)$ # Sample action from the policy
    $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ # Sample transition from the environment
    $D \leftarrow D \cup \{(s_t, a_t, R(s_t, a_t)), s_{t+1}\}$ # Store the transition in the dataset
  end for
  for each gradient step **do**
    $\omega_i \leftarrow \omega_i - \lambda_\omega \nabla_{\omega_i} L_{\omega_i}$ for $i \in \{1, 2\}$ # Update the Q function parameters
    $\sigma \leftarrow \sigma - \lambda_\sigma \nabla_\sigma L_\sigma$ # Update policy weights
    $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha L_\alpha$ # Update temperature
    $\overline{\omega_i} \leftarrow \tau\omega_i + (1-\tau)\overline{\omega_i}$ for $i \in \{1, 2\}$ # Update frozen Q-network weights
  end for
Output $\sigma, \omega_1, \omega_2$

In Algorithm 1, the L_σ, L_ω, and L_α are the loss function of training policy function, Q-value functions and the temperature factor parameterized by σ,ω and α, respectively. λ_σ , λ_ω and λ_α are learning rate of parameters in policy function, Q function and temperature function. According to [41], we have

$$L_\sigma = E_{s_t \sim D}\left[E_{(s_t, a_t) \sim \pi_\sigma}[\alpha \log P_\sigma(a_t|s_t) - Q_\omega(s_t, a_t)]\right], \tag{6}$$

$$L_\omega = E_{(s_t, a_t) \sim D}\left[\frac{1}{2}\left(Q_\omega(s_t, a_t) - \left(r(s_t, a_t) + \gamma E_{s_{t+1}}[V_{\overline{\omega}}(s_{t+1})]\right)\right)^2\right] \tag{7}$$

$$L_\alpha = E_{(s_t, a_t) \sim \pi_\sigma}[-\alpha \log P_\sigma(a_t|s_t) + \overline{H}] \tag{8}$$

The $V_{\overline{\omega}}(s_{t+1})$ in (7) is the soft state value, and can be calculated as:

$$V(s_t) = E_{(s_t, a_t) \sim \pi_\sigma}[Q_{\overline{\omega}}(s_t, a_t) - \alpha \log P_\sigma(a_t|s_t)]. \tag{9}$$

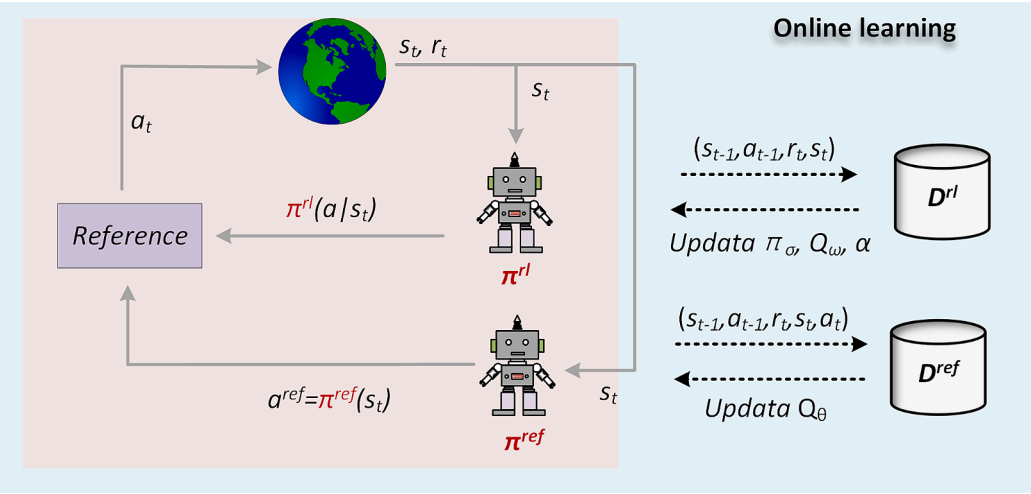

Fig. 1. The Framework of reference RL.

More details about the SAC algorithm and its theoretical foundations can be found in [41].

4. Reference RL based traffic signal control

This section first provides a detailed description of the reference RL, followed by its application to the TSC problem.

4.1. The reference RL

Compared with classic RL algorithms, the reference RL incorporates a reference policy π^{ref} to provide reference actions a^{ref} to assist the agent (π^{rl}) during the training process, thereby quickly upgrading system performance early on. Fig. 1 illustrates the framework of the reference RL algorithm.

At each time step t, the agent observes the environment to obtain the state representation s_t, and generates an action distribution $\pi^{rl}(.|s_t)$ based on its current policy π^{rl}. Simultaneously, the reference policy π^{ref} provides a reference action $a_t^{ref} \leftarrow \pi^{ref}(s_t)$ based on the same state s_t. The *Reference* procedure presented in Fig. 1 requires the agent sample an action a_t^{rl} from the action distribution $a_t^{rl} \sim \pi^{rl}(.|s_t)$. If the Q-value constraint in (10) is not met, a new a_t^{rl} should be sampled.

$$Q_\theta(s_t, a_t^{rl}) \geqslant Q_\theta(s_t, a_t^{ref}) \tag{10}$$

Separate from the Q functions used in the SAC algorithm, the $Q_\theta(.)$ is an additional value function that evaluates both the agent's action and the reference action. A maximum number of resampling attempts η is set to prevent excessive resampling. If the constraint (10) can be satisfied within η attempts, the resampled action from the agent a_t^{rl} will be executed $a_t = a_t^{rl}$. Otherwise, the reference action will be adopted $a_t = a_t^{ref}$.

SAC algorithm presented in *Section 2* is adopted to train the agent due to its stability and exploration efficiency in continuous action spaces. The SAC optimizes a stochastic policy by balancing expected return and entropy, encouraging exploration in complex environments like traffic system. The parameters involved in the SAC (i.e. σ, ω, and α) are updated using gradient decent with loss functions (6)-(8). The Q-value function Q_θ introduced in (10) is independent of the Q networks involved in the SAC algorithm. Based on multiple experimental comparisons, we found that adopting a separate Q network can achieve more stable performance improvements compared to utilizing the Q-value function Q_ω of SAC algorithm. Therefore, an independent Q network Q_θ is employed to impose a Q-value constraint during action selection. The function Q_θ is updated alongside the SAC algorithm, with the following loss function:

$$L_\theta = E_{(s_t, a_t, s_{t+1}, a_{t+1}) \sim D^{ref}} \left[\frac{1}{2} (Q_\theta(s_t, a_t) - (r_{t+1} + \gamma Q_{\bar{\theta}}(s_{t+1}, a_{t+1})))^2 \right] \tag{11}$$

where $Q_{\bar{\theta}}$ is a frozen recent model of Q_θ. It is worth noting that the samples in data buffer D^{rl} and D^{ref} are different shaped. In the SAC algorithm, the soft Q-value Q_ω is computed iteratively, where the policy is involved in to provide the probability distribution of actions as shown in (7) and (9). However, the Q_θ is updated based on the experienced trajectory data. Therefore, only the actual executed actions both a_t and a_{t+1} stored in the D^{ref} are used to approximate the Q values. The online learning process of the reference RL al-

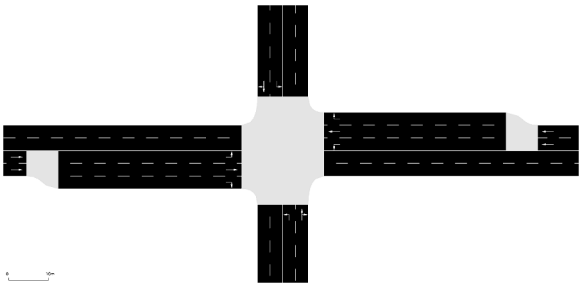

Fig. 2. The layout a classic four-leg intersection.

gorithm is presented in Algorithm 2.

Algorithm 2: The online learning algorithm

Input and initialization:
$\theta_1, \theta_2, \overline{\theta_1} \leftarrow \theta_1, \overline{\theta_2} \leftarrow \theta_2$ # Initialize $Q_{\theta_1}, Q_{\theta_2}, Q_{\overline{\theta_1}}, Q_{\overline{\theta_2}}$ with random parameters
σ, $\alpha, \omega_1, \omega_2, \overline{\omega_1} \leftarrow \omega_1, \overline{\omega_2} \leftarrow \omega_2$ # Initialize the parameters of the SAC algorithm
$D^{ref} \leftarrow \varnothing, D^{rl} \leftarrow \varnothing$ # Initialize the database of reference policy and agent
$\lambda_\theta, \lambda_\sigma, \lambda_\omega, \lambda_\alpha, \eta$# Initialize the learning rates and resample limits
For each episode **do**
  For each environment step **do**
    $\pi^{rl}(.|s_t)$# Generate the action distribution based on the agents' policy
    $a_t^{ref} \leftarrow \pi^{ref}(s_t)$# Get reference action from the reference policy
    $a_t = REFERENCE\left(a_t^{ref}, \pi^{rl}(.|s_t)\right)$# Execute the reference process
    $D^{ref} \leftarrow D^{ref} \cup \{(s_{t-1}, a_{t-1}, r_t, s_t, a_t)\}$ # Store the transition to D^{ref}
    $D^{rl} \leftarrow D^{rl} \cup \{(s_{t-1}, a_{t-1}, r_t, s_t)\}$ # Store the transition to D^{rl}
If TIME FOR TRAINING:
    For each gradient step **do**
      $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta L_\theta$ for $\theta \in \{\theta_1, \theta_2\}$ # Update $Q_{\theta_1}, Q_{\theta_2}$ with data in D^{ref}
      $\omega \leftarrow \omega - \lambda_\omega \nabla_\omega L_\omega$ for $\omega \in \{\omega_1, \omega_2\}$ # Update $Q_{\omega_1}, Q_{\omega_2}$ with data in D^{rl}
      $\sigma \leftarrow \sigma - \lambda_\sigma \nabla_\sigma L_\sigma$ # Update the policy parameters with data in D^{rl}
      $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha L_\alpha$# Update the temperature parameter with data in D^{rl}
    End for
$\overline{\theta_i} \leftarrow \tau\theta_i + (1-\tau)\overline{\theta_i}$ for $i \in \{1,2\}$ # Update frozen weights $Q_{\overline{\theta_1}}, Q_{\overline{\theta_2}}$
$\overline{\omega_i} \leftarrow \tau\omega_i + (1-\tau)\overline{\omega_i}$ for $i \in \{1,2\}$ # Update frozen weights $Q_{\overline{\omega_1}}, Q_{\overline{\omega_2}}$
End for
End for
Output $\theta_1, \theta_2, \omega_1, \omega_2, \sigma, \alpha$

The L_σ, L_ω, L_α and L_θ can refer to (6), (7), (8) and (11) respectively. The *REFERENCE* procedure is as follow:

The *REFERENCE* process

Input: $a_t^{ref}, \pi^{rl}(.|s_t), \eta$
Initialize: $i = 1, a_t^{rl} \sim \pi^{rl}(.|s_t)$
While $Q_\theta(s_t, a_t^{rl}) < Q_\theta\left(s_t, a_t^{ref}\right)$ and $i \le \eta$: # If Q value constraint is not satisfied
  $a_t^{rl} \sim \pi^{rl}(.|s_t)$ # Resample a_t^{rl} from the action distribution
  $i \leftarrow i + 1$
End While
If $i > \eta$: # If the Q value constraint cannot be satisfied after resampling η times
  Output a_t^{ref} # Output a_t^{ref} as the next action
Else:
  Output a_t^{rl} # Output a_t^{rl} as the next action

In the REFERENCE procedure, Two Q networks with different initialized parameters, denoted as θ_1 and θ_2 respectively, are adopted to obtain better estimation of Q values, i.e. $Q_\theta(s_t, a_t) = \left(Q_{\theta_1}(s_t, a_t) + Q_{\theta_2}(s_t, a_t)\right)/2$.

4.2. The reference RL based traffic signal controller

This section details the design of the reference RL-based TSC model, specifically its state representation, action set, and reward function for solving the TSC problem.

4.2.1. The TSC problem

The core of TSC is to manage traffic flows by assigning right-of-way to different directions at an intersection, thereby preventing collisions and ensuring smooth traffic movement. Fig. 2 illustrates the layout of a classic four-leg intersection.

To help drivers anticipate signal changes, a fixed-sequence, four-phase signal scheme is adopted, consisting of: East-West through phase, East-West left turn phase, North-South through phase and North-South left turn phase. Within the RL framework, agents are tasked with observing the traffic state before the start of each phase and determining the green duration of the next phase. forgiven the continuous action space in TSC problems, SAC algorithm is adopted to train TSC agents.

This study aims to address the unsatisfactory system performance observed during the training process, which is primarily due to the immature signal control policies at the early stages of agent learning. This poses a challenge for real-world training of agents.

4.2.2. The design of the reference RL based TSC model

To accommodate traffic networks of varying scales, a distributed control framework is adopted, providing greater flexibility and adaptability. Each intersection in the road network is managed by an individual agent. From a practical traffic management perspective, the proposed TSC model relies on widely available traffic data that can be collected through inductive loops or camera-based techniques. In addition, the commonly-deployed fixed-sequence phase scheme is employed, where each phase allows specific traffic movements to enter the intersection simultaneously. The agent's task is to adjust the green duration at the start of the phase. The goal of the TSC controller is to minimize the total vehicle delay in the road network by dynamically adjusting phase durations in real time.

Within the RL framework, the state representation, action set and reward function in TSC problems need to be defined appropriately.

(1) State representation

For each intersection, the traffic in both inbound and outbound lanes are considered. Let L_{in} be the set of all lanes leading to the intersection, representing future traffic demand of movements. L_{out} is the set of downstream lanes, traffic on which indicates remaining storage capacity. At each time step t, the agent collects the number of queuing vehicles q_t^l(veh) and the waiting time w_t^l of the leading car for each lane l in $L_{in} \cup L_{out}$. In addition, the one-hot encoding of the next phase p_t^i is included in the state representation s_t. Therefore, at time step t, the traffic state observed by the agent i can be represented as:

$$s_t^i = \{q_t^l, w_t^l, p_t^i\}, \forall l \in L_{in}^i \cup L_{out}^i \tag{12}$$

The dimensionality of the state vector s_t^i depends on the number of incoming lanes $|L_{in}^i|$, the number of outgoing lanes $|L_{out}^i|$ of the intersection i and the number of phases $|p^i|$: $|s^i| = |L_{in}^i| + |L_{out}^i| + |p^i|$.

(2) Action set

Three common methods of defining actions include: changing the order of phases while keeping the phase duration fixed, altering the phase duration while maintaining the phase order, and adjusting both the phase order and duration simultaneously. This study adopts the second way, enabling the agent to control the green duration of phases while maintaining a fixed-sequence phase switching. The phase duration encompasses green time, yellow change and red clearance intervals. The green time must remain within a defined range $[g_{\min}, g_{\max}]$. Therefore, the action space is $A = [g_{\min}, g_{\max}]$ and $|A| = 1$.

(3) Reward function

The goal of traffic signal control is to minimize the delay of vehicles in the road network. In this study, the reward function R is defined as the weighted sum of the cumulative delay D and the throughput V at the intersection. The cumulative delay and the throughput of the intersection at time step t is defined as follows:

$$D_t = \sum_{k=t-1}^{t} \sum_{l \in L_{in}} q_k^l \tag{13}$$

$$V_t = \sum_{k=t-1}^{t} \sum_{l \in L_{in}} v_k^l \tag{14}$$

where q_k^l is the number of queuing vehicles at lane l in the time step k. The v_k^l is count of vehicles crossing the stop line in the k^{th} second. It is worth noting that the seconds encompassed in the interval [t-1, t] varies depending on the phase duration generated at t-1 step, the yellow and all-red intervals. The positive throughput variable is introduced to prevent the agent from getting negative feedback all the time and reinforcing the policy in a negative way. At time step t, the reward returned to the agent is:

$$r_t = \partial(\beta_1 D_t + \beta_2 V_t), \tag{15}$$

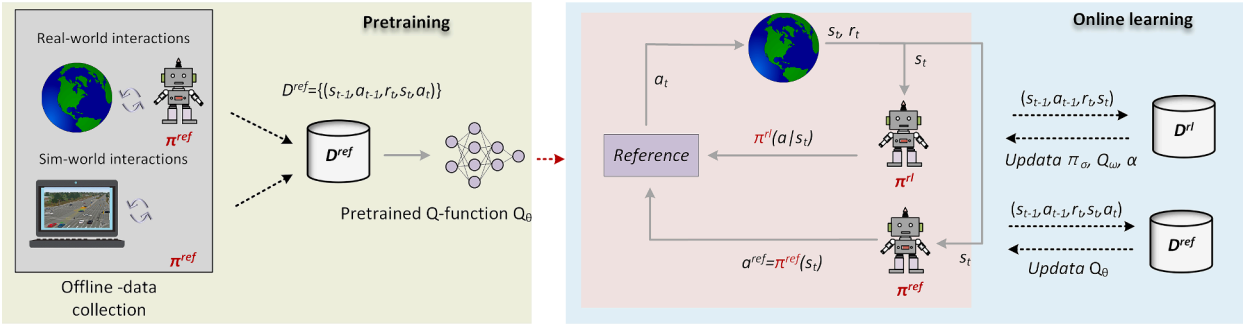

Fig. 3. Framework of the pretrained reference RL.

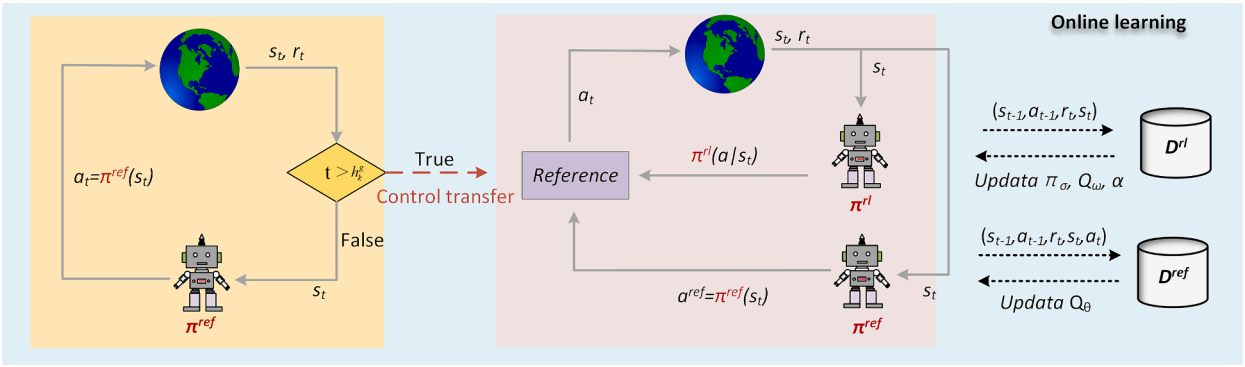

Fig. 4. Framework of the reference RL with jump start.

where β_1, β_2 are the weight factors of different parts in the reward function; ∂ is the regulation factor of reward.

We would like to clarify that the primary focus of this study does not lie in the design of communication mechanisms among agents. However, potential communications can be achieved expanding the dimension of state representation to encompass the state variables and action footprint of adjacent intersections [42]. Optimizing the reward signals of neighboring agents in conjunction with the reward of the ego agent can facilitate the realization of cooperative actions among agents [42,43]. Additionally, other studies have introduced the attention mechanism to enhance communication among agents [44]. More communication and cooperation strategies can refer to [45].

5. Extensions of the reference RL

The reference RL discussed in *Section 4* starts with random initialized Q networks, which inevitably leads to inaccurate value estimation o of both the agent's action and the reference actions at the beginning of training. This section presents two optional extensions of the reference RL method, aiming to enhance system performance of the initial few training episodes to ensure greater safety and efficiency in the real-world agent learning.

5.1. Pretraining with offline data

Pretraining with offline data can reduce the occurrence of poor timing plans resulting from inaccurate Q-value estimates. However, as noted by [31], actor-critic methods can hardly benefit from a well initialized Q network due to the bootstrap error. Therefore, rather than pretraining the Q-value functions contained in the SAC algorithm (Q_ω), the reference Q-value function (Q_θ) is initialized using the parameters pretrained from offline dataset. This extension of reference RL is termed the "pretrained reference RL", and its framework is presented in Fig. 3.

With offline data collected from either simulation environment or real-world system (if a trusted reference policy is provided), the reference Q-value function Q_θ can be pretrained using the loss function (11).

5.2. Cooperate with the jump-start reinforcement learning algorithm (JSRL)

The JSRL algorithm, proposed by [4], gradually transfers control from a guide-policy to an exploration-policy (the policy of agent) during the training process. In the k^{th} training episode, the guide policy controls the system for the first h_k^g steps, leading the agent into

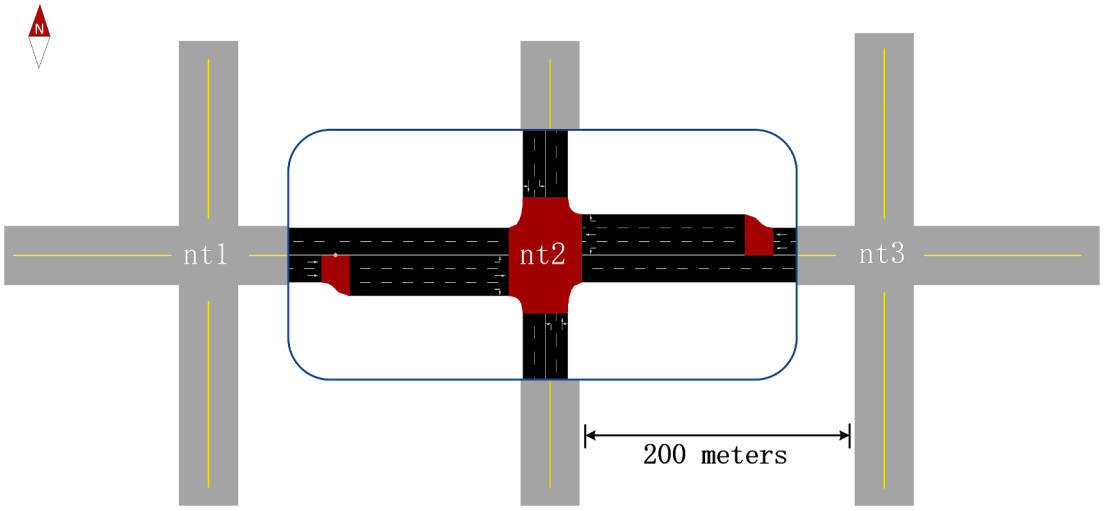

Fig. 5. The corridor scenario.

"good" states. The exploration policy then takes over the control, allowing the agent to interacts with the environment. As the exploration policy improves over time, the agent progressively takes control earlier and earlier. Incorporating the JSRL algorithm into the reference RL mitigates the impact of inaccurate reference Q-value function during early training episodes. This combined approach is referred to as "reference JSRL", as shown in Fig. 4.

In the JSRL, a curriculum strategy is used to reduce the guide steps h_k^g as training progressing. Intuitively, the JSRL starts with a large guide-step $h_1^g = \overline{h}$ at the first training episode, and then decrease the guide-step as the agent's moving average performance improves over several episodes. However, manually determining the optimal horizon of the moving average is impractical in real-world training. A short period may result in the reference policy relinquishing control too soon, while a long period may hinder the agent's learning due to limited control opportunities. To address this issue, we implemented a modified curriculum strategy to decrease the guide step gradually. Specifically, at the k^{th} training episode, if the score of the episode surpasses the reference score, the guide step h_k^g is reduced by a fixed number of seconds Δh in subsequent episodes until the agent fully takes control ($h_k^g = 0$).

6. Numerical results

In real-world learning scenarios, maintaining a reasonable level of performance throughout the agent's learning process is critical, especially in the early training stages when the agent's policy is underdeveloped and prone t undesirable actions. This section presents numerical experiments designed to validate the advantage of the reference-RL in improving system performance during the agent's learning process, particularly at the initial training stages. This section first describes the simulated traffic scenarios and several reference timing policies. Then, the reference-RL is evaluated and compared with the MA2C method [42] and the jump-start reinforcement learning (JSRL) reported in [4]. At last, two extension versions of the reference RL are examined and analyzed to assess their effectiveness in enhancing the system performance during the initial training stages.

6.1. Simulation settings

Numerical experiments were performed on the SUMO [38] platform. SUMO is one of the most frequently used, open-source traffic simulation packages, which provides the execution and evaluation of traffic simulations. Given considerations like traffic safety, efficiency, and other concerns expressed by the traffic management department, carrying out field experiments is a challenging task. Therefore, the SUMO platform serves as a proxy for real traffic system to test the performance of the reference RL in our experiment.

A corridor network with three signalized intersections and a real-world network with seven signalized intersections are constructed in the SUMO platform to evaluate the performance of algorithms.

6.1.1. The corridor network

The simulated corridor network is shown in the Fig. 5, where the west-east roads are considered as the main roads, while the north-east roads are designated as branch roads. The west-east roads feature dedicated lanes for each traffic movement, ensuring separate lanes for different directions of travel. The north–south roads have a shared turning lane for right turns and straight movements. The traffic demand data of different movements in intersections is provided in Appendix A.1.

At the start of each phase, the agent observes the traffic state and generates action $a_t \in [-1,1]$, which would be linearly mapped to an integer value between the minimum $g_{\min} = 5s$ and the maximum $g_{\max} = 120s$. Each simulation episode starts with an empty network, and lasts 3600 s.

Fig. 6. The real-world scenario.

Table 1
The fixed signal plan of intersections.

Phases		East-West through phase	East-West left turn phase	North-South through phase	North-South left turn phase
The corridor network					
Green duration(s)	nt1	120	32	27	40
	nt2	110	12	23	19
	nt3	120	28	44	40
The real-world network					
Green duration(s)	nt1	26	27	13	30
	nt3	45	19	13	19
	nt4	28	5	7	5
	nt5	30	9	14	19
	nt7	14	19	14	27
	nt9	19	34	11	13
	nt11	17	23	9	23

6.1.2. *The real-world network*

Fig. 6 illustrates the studied area of Yangzhou City, where signalized intersections are marked by ★ and other intersections are priority-controlled intersections. Most right-turn traffic travels through channelized lanes, making it exempt from signal control. The first horizontal road from north to south and the second vertical road from west to east are primary roads, while the remaining roads are minor arterials or collectors.

In this simulation, phase durations are constrained within the range of 5 s to 90 s. The traffic demand of the studied area is presented in Appendix A.2. Each training episode lasts 3600 s.

6.2. *Reference policies and benchmarks*

6.2.1. *Reference policies*

The reference RL can work with any reliable TSC methods. In this study, we don't intend to make much effort to the design of reference signal control policy. While it is true that more sophisticated control strategies may provide better suggestions on agent's decision makings, their implementation often require extensive fine-tuning in real-world practice to achieve optimal performance. In addition, the choice of a simple reference policy aligns with the prevailing practices in real-world traffic management systems. Therefore, we decide to adopt two well-known adaptive TSC models: adaptive-Webster's controller [46] and cycle phase BackPressure controller [47], as well as a fixed-time signal plan.

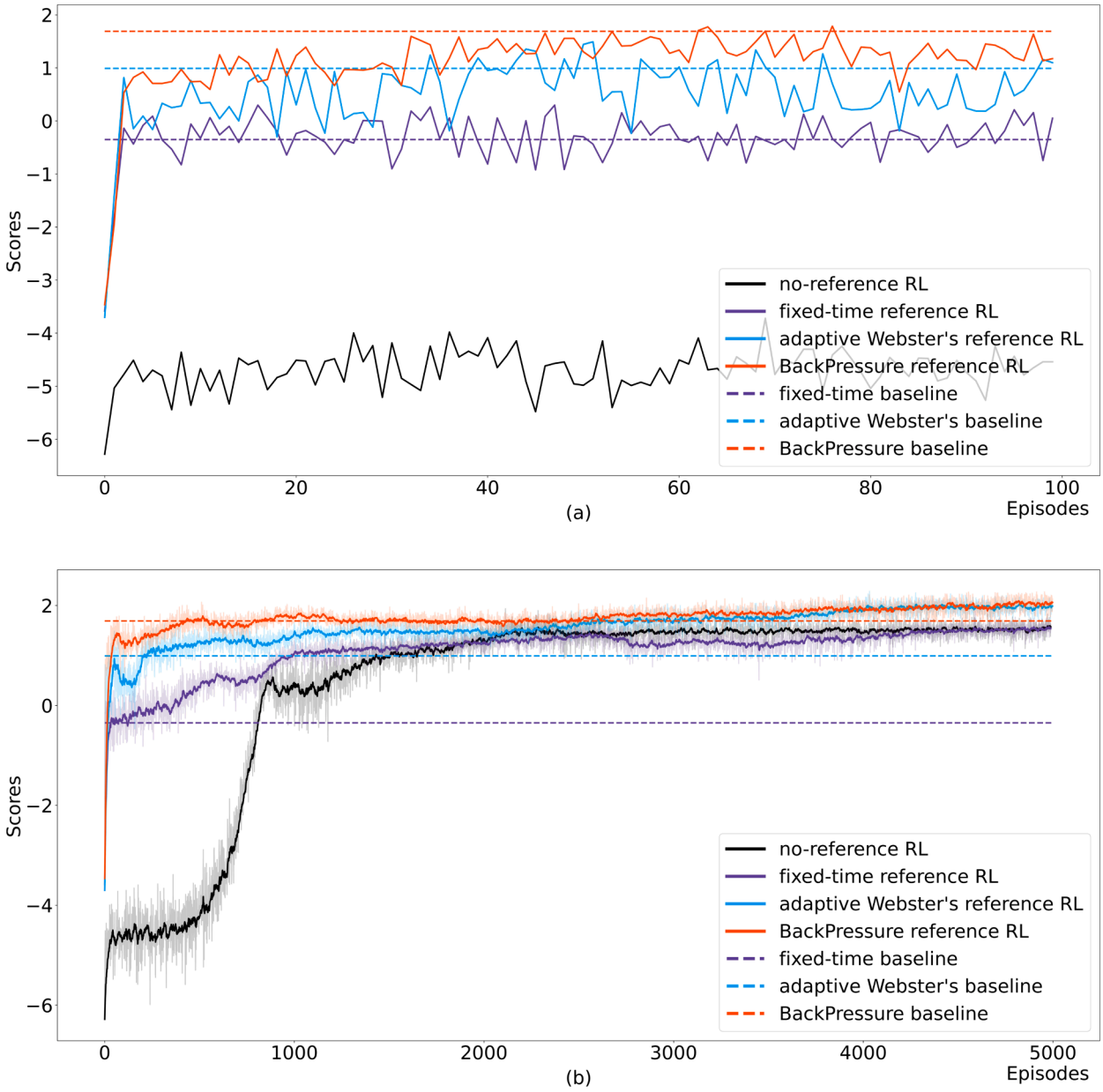

Fig. 7. The training curves of algorithms in the corridor network. (a) The first 100 episodes; (b) The whole training process.

(1) The fixed-time policy. It is a static timing plan regardless of system state. The fixed signal plans for the corridor network and the real-world scenario are showed in Table 1, calculated using Webster's formula. The yellow change and red clearance time between phases are set to 2 s and 3 s, respectively. Meanwhile, overlong phase durations are truncated within the green duration constraint, such as the East-West through phase in the corridor network.

(2) The adaptive Webster's policy. The adaptive Webster's policy estimates traffic volumes based on the traffic data collected over a recent interval W. Then, as reported in [48], the Webster's method is employed to calculate the cycle length and phase durations for the subsequent duration of W. For algorithm details, please refer to [46].

(3) The cyclic phase BackPressure policy. The BackPressure method works by dynamically adjusting traffic signal timings based on "pressure" values of phases, which reflect the difference between incoming and outgoing traffic. The phase with the highest pressure is given the right of way during the next period. The cyclic phase BackPressure policy proposed in [47] addressed two weaknesses in the previous BackPressure method [49]. First, it ensures a predictable and safe signal sequence. Second, it addressed the challenge of estimating "turning fractions" of traffic at intersections, which is crucial for previous BackPressure

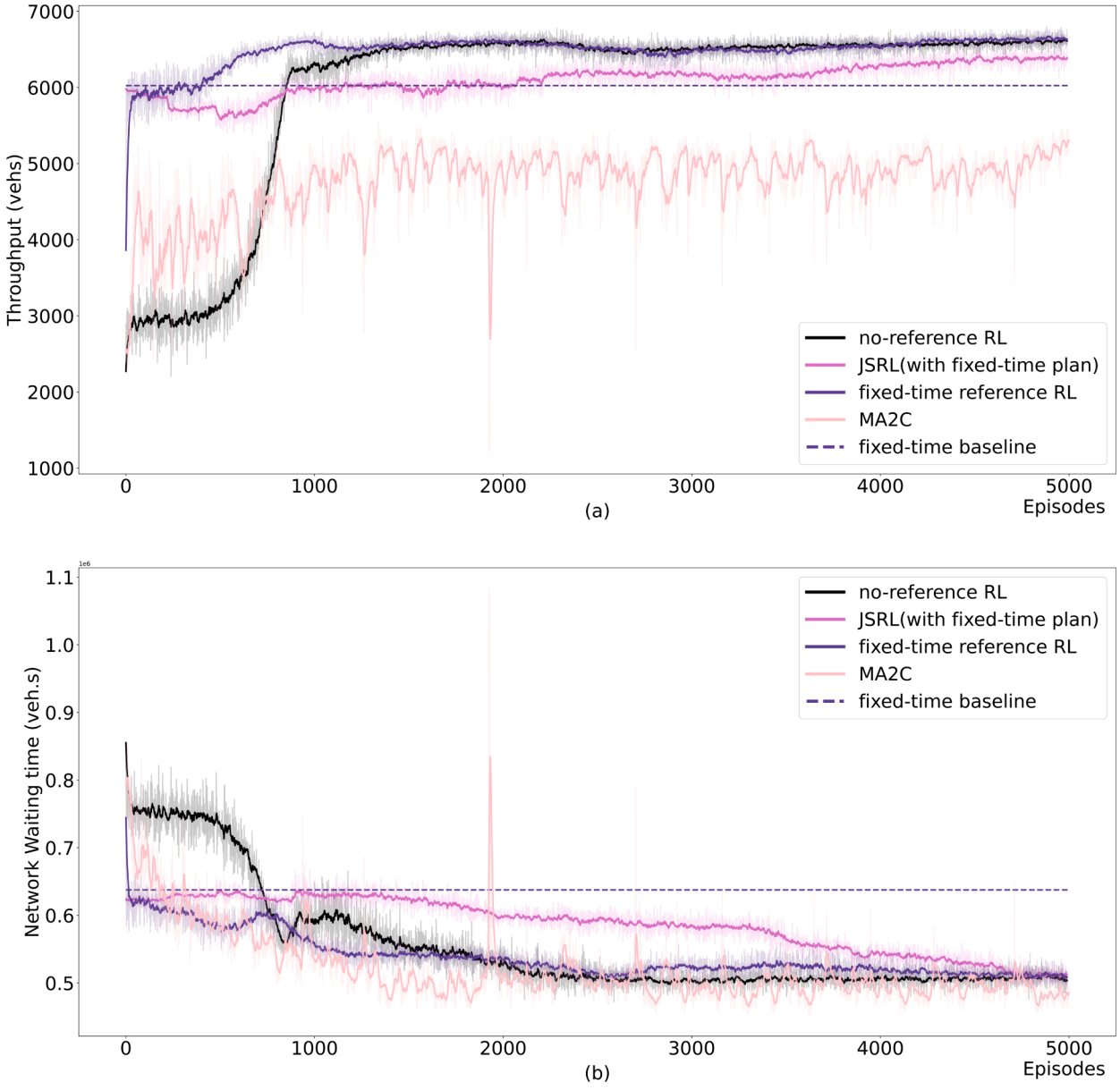

Fig. 8. The progress of evaluation indicators of the fixed-time reference RL. (a) The network throughput; (b) The network waiting time.

method but is difficult to estimate accurately. However, since the cyclic phase BackPressure method only optimizes the green time splits of phases (please refer to [47]), a fixed cycle length of 120 s is adopted in our experiments.

6.2.2. *Benchmarks and simulation settings*

We compare the proposed Reference-RL method with other two state-of-the-art methods, i.e. Jump-Start Reinforcement Learning (JSRL) [4] and the multi-agent A2C (MA2C) [42]. The JSRL adopts a guide policy that progressively hands control to the RL agent, accelerating the early training process. The JSRL would adopt the same guide policies as the reference RL. The MA2C incorporates traffic information from neighboring intersections into its state representation and reward function. A spatial discount factor is adopted to scale down the reward signal of neighboring agents to make the agent focus more on local traffic. The MA2C promotes communication among agents, thereby improving the robustness and overall performance of the TSC model.

Additionally, detailed setting about the super-parameters and DNN network for the reference-RL, the JSRL and the MA2C are presented in Appendix B.

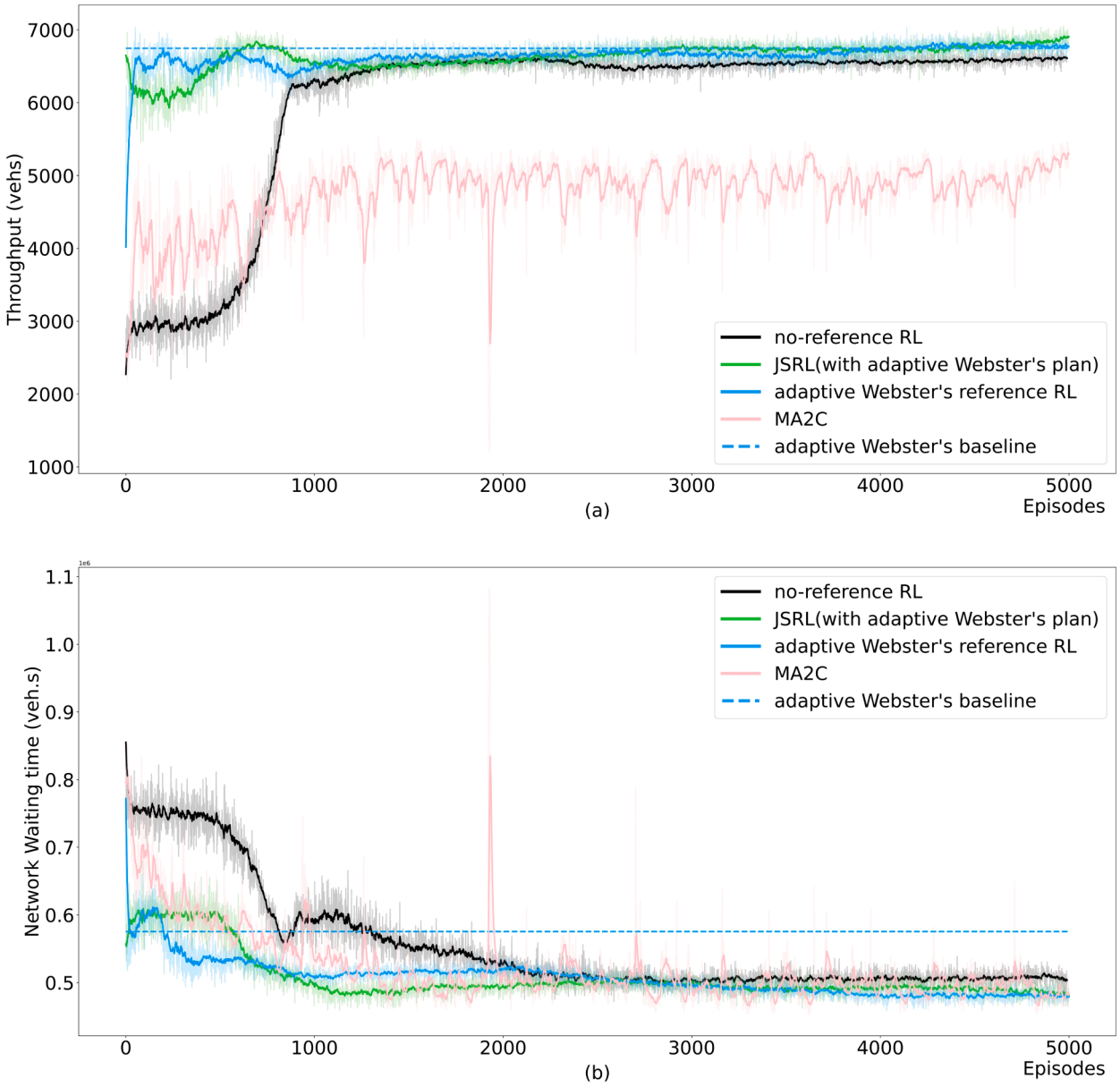

Fig. 9. The progress of evaluation indicators of the adaptive Webster's reference RL. (a) The network throughput; (b) The network waiting time.

6.3. Performance of the reference RL

Three reference policies are integrated with the reference RL: the fixed-time policy (fixed-time reference RL), the adaptive Webster's policy (adaptive Webster's reference RL), and the BackPressure policy (BackPressure reference RL). These combinations allow evaluating and comparing the influence of different reference policies on the agent's learning process. The method without the reference mechanism is referred to as "no-reference RL".

6.3.1. Corridor network

Fig. 7 plots the scores of training episodes, providing insights into the progress of agent's learning. The dashed lines in Fig. 7 represent the average scores of the three reference policies over 20 tests. Therefore, these dash lines are straight throughout the training process.

The Fig. 7 (a) shows that all three reference RL methods significantly outperform the no-reference RL method during the first 100 episodes of training. The fixed- time reference RL rapidly achieves comparable performance to the fixed-time baseline after just a few training episodes. Compared with the fixed-time reference RL, the adaptive Webster's reference RL and the BackPressure reference RL

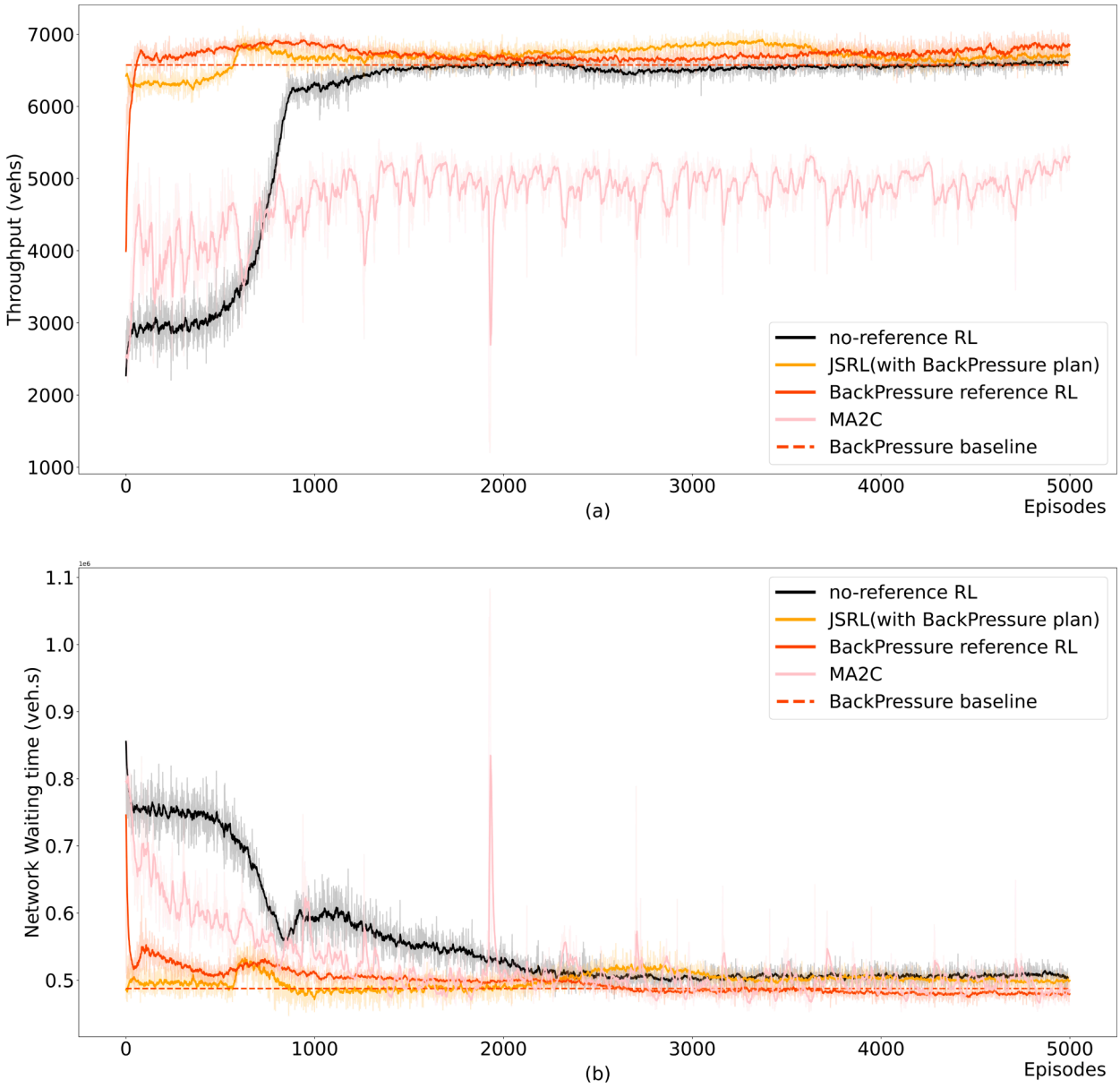

Fig. 10. The progress of evaluation indicators of the BackPressure reference RL. (a) The network throughput; (b) The network waiting time.

provides even greater promotion on the initial learning performance of agents, but take longer to surpass their respective baselines. Among the three, the BackPressure reference RL exhibits the best performance boost. The results demonstrate that the reference RL approach can significantly enhance the learning performance of the agent during the early training stages. Different reference strategies promote the training process to varying extents. While a well-performing policy can achieve greater promotion of initial training performance, it takes more time for the agent to exceed the policy's baseline performance.

The exploration of the state space by the agent in no-reference RL and fixed-time reference RL is presented in Appendix C, aiding in a more intuitive understanding of the reference mechanism.

Fig. 7 (b) exhibits the learning curves of agents over 5000 episodes. The plots reveal that the no reference RL converges after approximately 2500 training episodes. In contrast, it only takes 400 episodes (1000 episodes) for the BackPressure reference RL (the adaptive Webster's reference RL and) to reach the convergence score of no reference RL. The fixed-time reference RL initially improves faster than the no-reference RL but eventually converges at a similar performance level. The adaptive Webster's reference RL and the BackPressure reference RL achieve greater performance promotion at the initial stages and higher convergence scores than the no-reference RL.

We take the network throughput and the total waiting time as metrics to evaluate the performance of algorithms. Figs. 8-10 displays

Table 2
Numerical results of algorithms in corridor scenario.

	The first 20 episodes		Convergence performance	
	Network throughput (veh)	Total waiting time(s)	Network throughput (veh)	Totalwaiting time (s)
no-reference RL	2818	773,042	6610	506,415
MA2C	2639	788,592	4820	504,713
JSRL (with fixed-time plan)	5947	621,883	6387	512,055
fixed-time reference RL	**5703**	**616,927**	**6640**	**500,778**
JSRL (with adaptive Webster's)	6486	579,972	6827	487,122
adaptive Webster's reference RL	**5664**	**561,540**	**6774**	**481,045**
JSRL (with BackPressure)	6354	495,084	6689	498,760
BackPressure reference RL	**5854**	**534,409**	**6810**	**480,138**

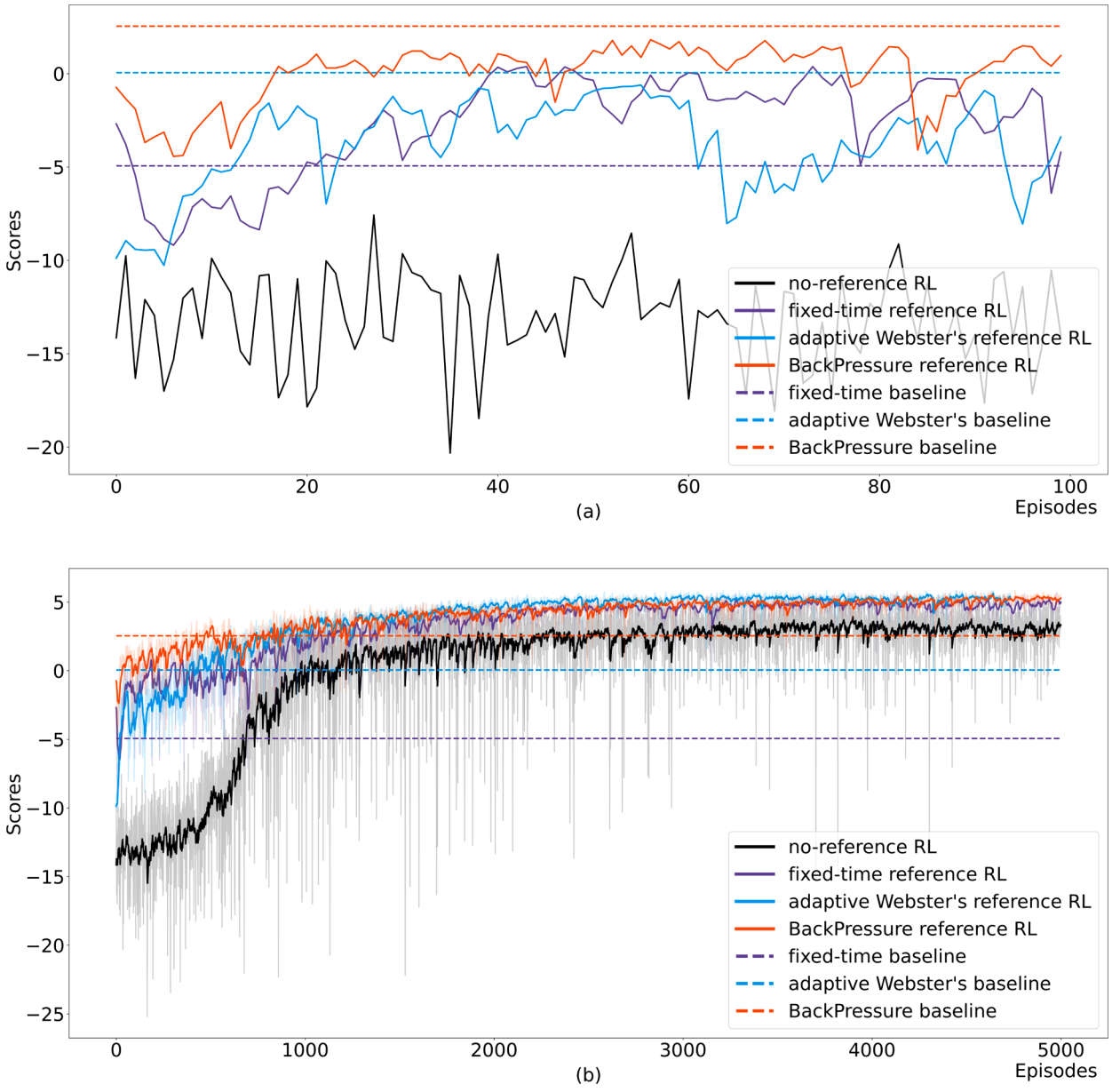

Fig. 11. The training curves of algorithms in the real-world scenario. (a) The first 100 episodes; (b) The whole training process.

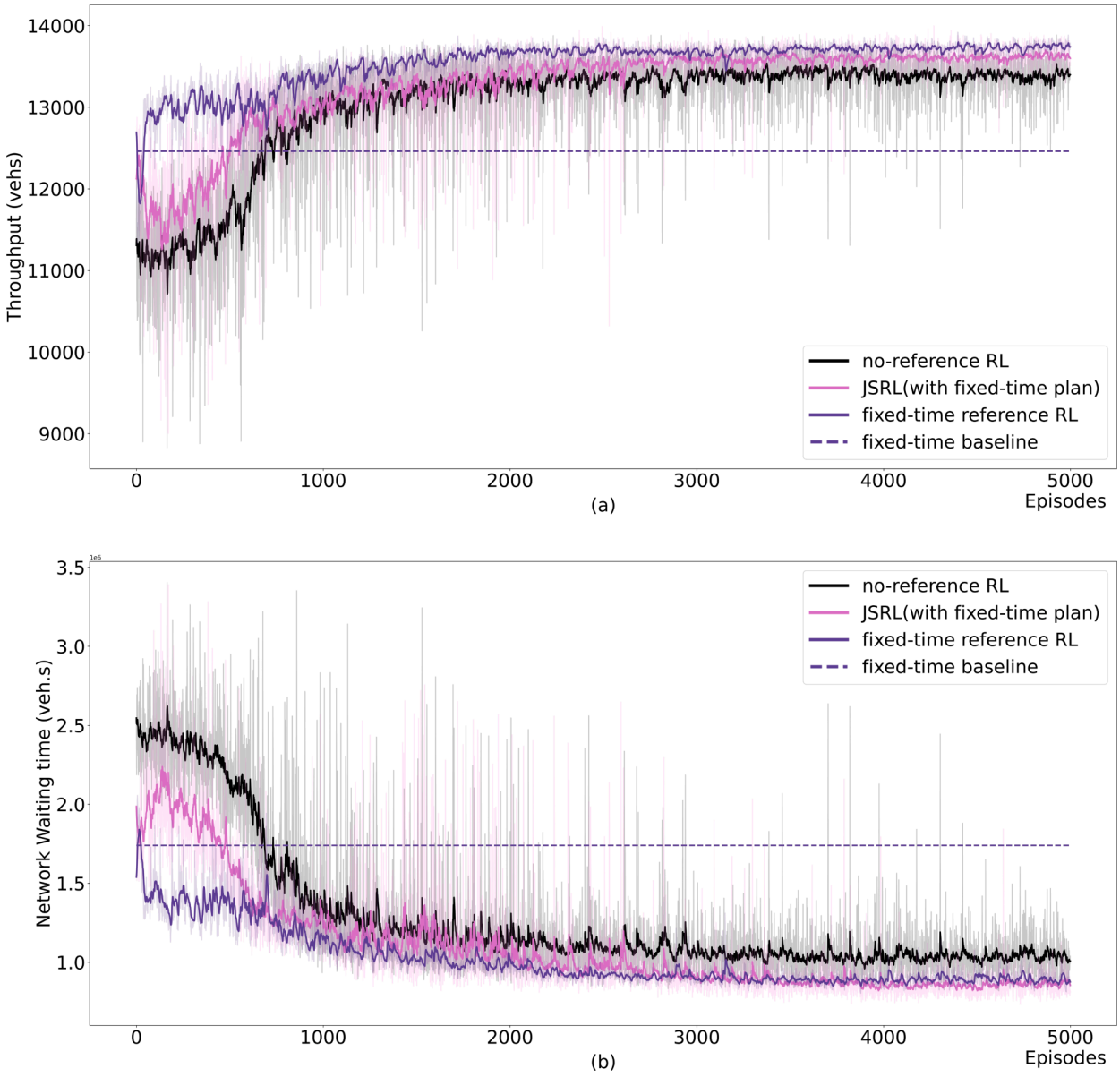

Fig. 12. The progress of evaluation indicators of the fixed-time reference RL in the real-world scenario. (a) The network throughput; (b) The network waiting time.

the progress of throughput and total waiting time of the network during the training process. The JSRL employs the fixed-time policy, the adaptive Webster's policy and the BackPressure policy as the agent's guidance policy, respectively. Additionally, Table 2 presents the numerical results of evaluations.

According to the plots illustrated in Figs. 8-10 and the numerical data shown in Table 2, the no-reference RL and all three reference RL methods achieve higher network throughput and less vehicle waiting time than MA2C during the first 20 training episodes. Comparing with the MA2C, the fixed-time reference RL, the adaptive Webster's reference RL and the BackPressure reference RL enhance the throughput by approximately 116.1 %, 114.6 % and 121.8 % and reduce the total waiting time by 21.8 %, 28.8 %, and 32.2 %, respectively. However, as for the convergence performance, the MA2C develops control policy with less throughput and less waiting time than no-reference RL and fixed-time reference RL. The reason for this result may come down to the different design of reward functions. The MA2C approach incorporates the queuing length and the cumulative delay of the first vehicle into the instant reward, instead of the changes in network throughput. The adaptive Webster's reference RL and the BackPressure reference RL surpass the performance of MA2C in both network throughput and vehicle waiting time.

The numerical data in Table 2 also indicate that all three reference RL perform slightly worse than their corresponding JSRL methods in the first 20 training episodes. However, after more training episodes, three reference RL approaches surpass their

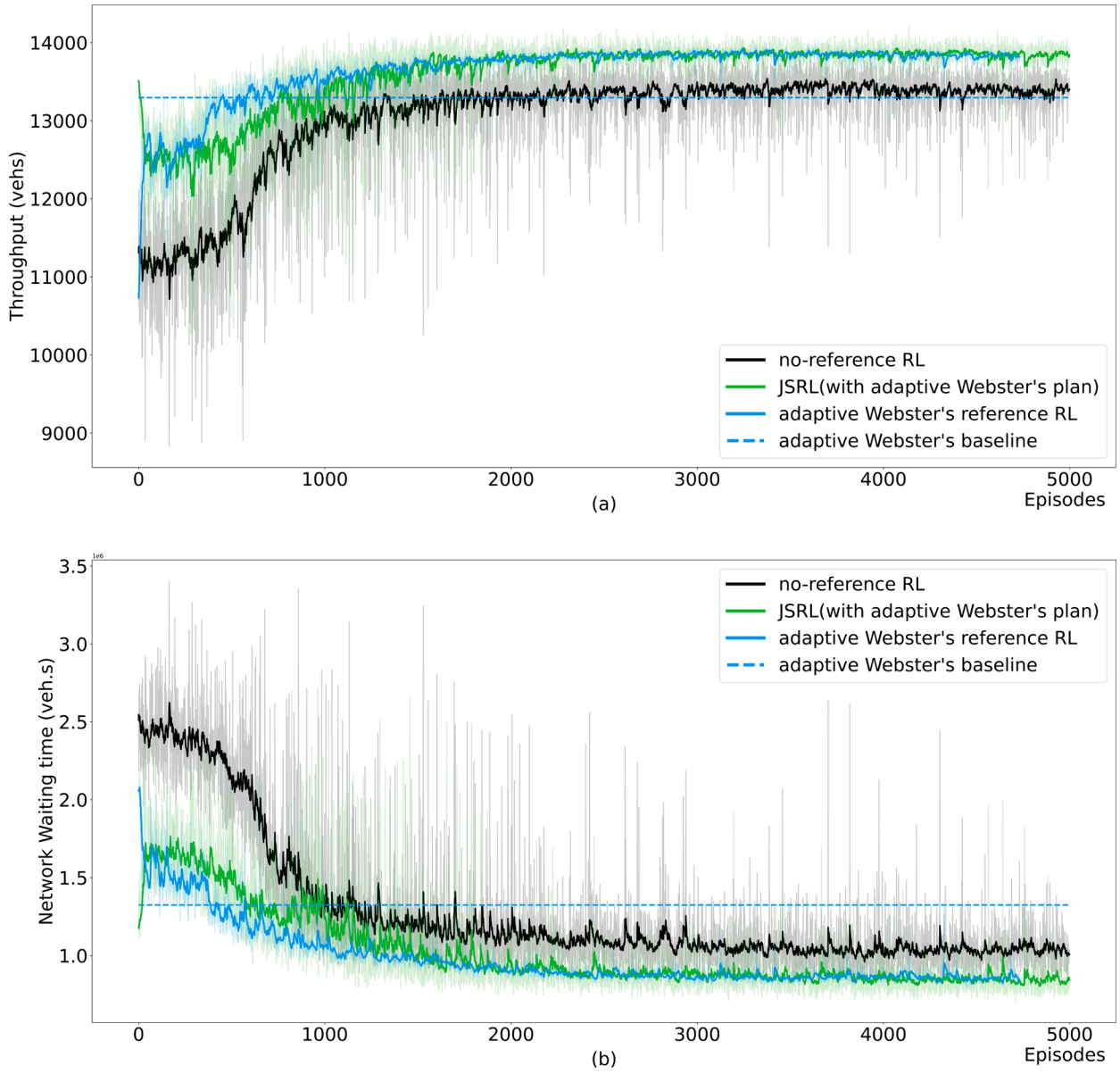

Fig. 13. The progress of evaluation indicators of the adaptive Webster's reference RL in the real-world scenario. (a) The network throughput; (b) The network waiting time.

corresponding JSRL methods. Specifically, the fixed-time reference RL reaches the performance of the JSRL (with fixed-time plan) at the 5th training episode, and surpass it after 200 training episodes. The adaptive Webster's reference RL and the BackPressure reference RL perform equivalent to their corresponding JSRL methods within 20 training episodes. The adaptive Webster's reference RL outperforms the JSRL (with adaptive Webster's plan) after approximately 150 training episodes, while the BackPressure reference RL surpasses the performance of the JSRL (with BackPressure plan) after 3000 episodes. In addition, as illustrated in Figs. 8-10, due to the artificially designed learning curriculum, JSRL methods exhibit obvious periodic performance drops during the training process.

The comparative analysis between the reference RL methods and the MA2C model elucidates notable advancements in network throughput and reduction in vehicle waiting time during the initial training stages. Despite the reference RL methods initially lag behind their JSRL counterparts, they quickly surpass them after a short training period. Furthermore, the reference RL methods demonstrate superior convergence performance in both increasing the network throughput and reducing vehicle waiting time.

6.3.2. Real-world network

Due to its minor enhancement in the agent's initial performance in the corridor scene, the MA2C method is excluded from the benchmarks in the subsequent real-world network experiments. Fig. 11 plots the training curves for algorithms in the real-world

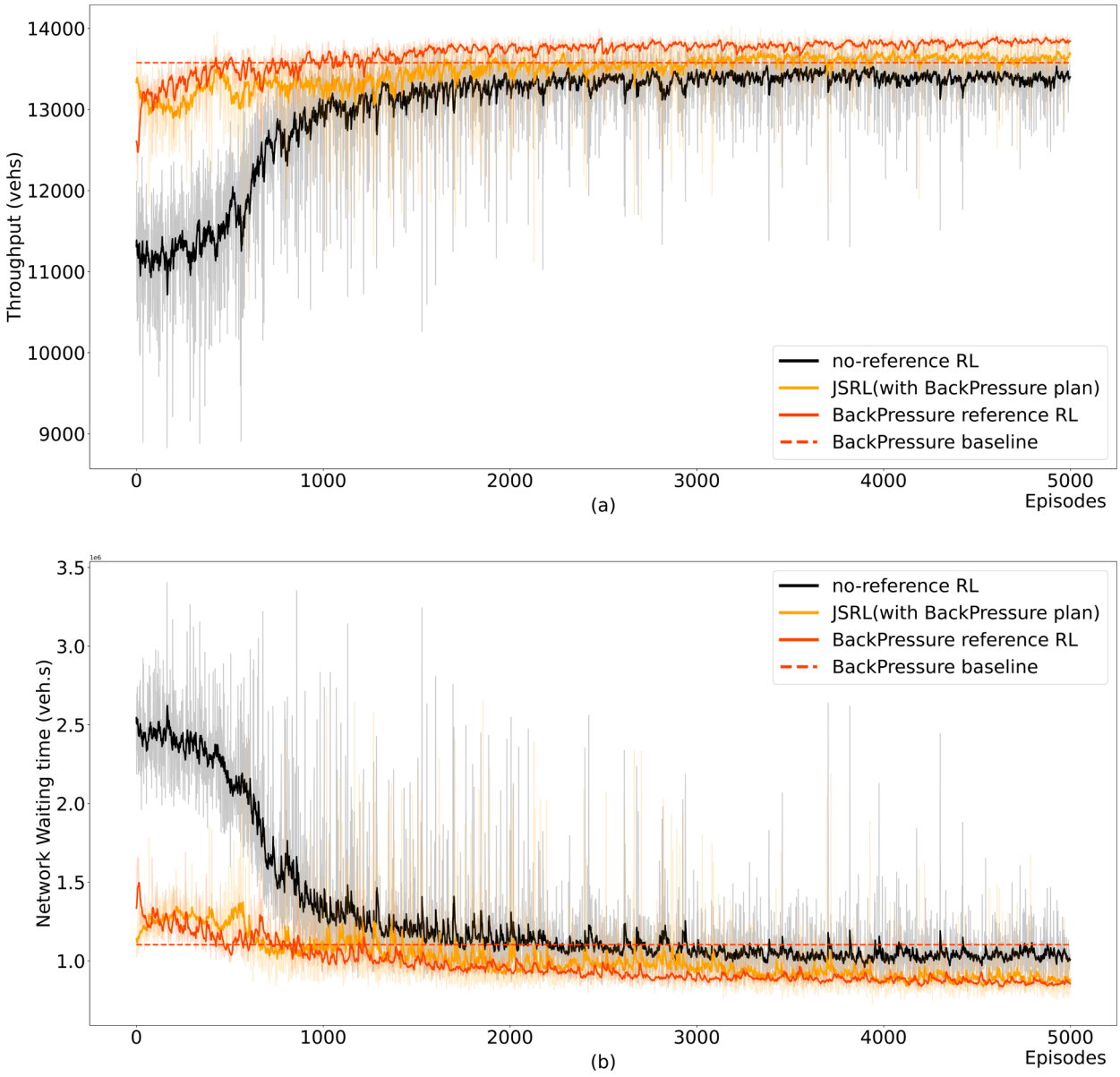

Fig. 14. The progress of evaluation indicators of the BackPressure reference RL in the real-world scenario. (a) The network throughput; (b) The network waiting time.

Table 3
Numerical results of algorithms in real-world scenario.

	The first 20 episodes		Convergence performance	
	Network throughput (veh)	Total waiting time(s)	Network throughput (veh)	Totalwaiting time (s)
no-reference RL	11,200	2,441,501	13,382	1,029,273
JSRL (with fixed-time plan)	12,387	1,720,937	13,632	862,885
fixed-time reference RL	**11,758**	**1,866,047**	**13,734**	**892,164**
JSRL (with adaptive Webster's)	13,117	1,310,974	13,818	860,698
adaptive Webster's reference RL	**12,036**	**1,798,859**	**13,812**	**860,615**
JSRL (with BackPressure)	13,232	1,158,170	13,638	889,054
BackPressure reference RL	**12,713**	**1,483,161**	**13,831**	**866,212**

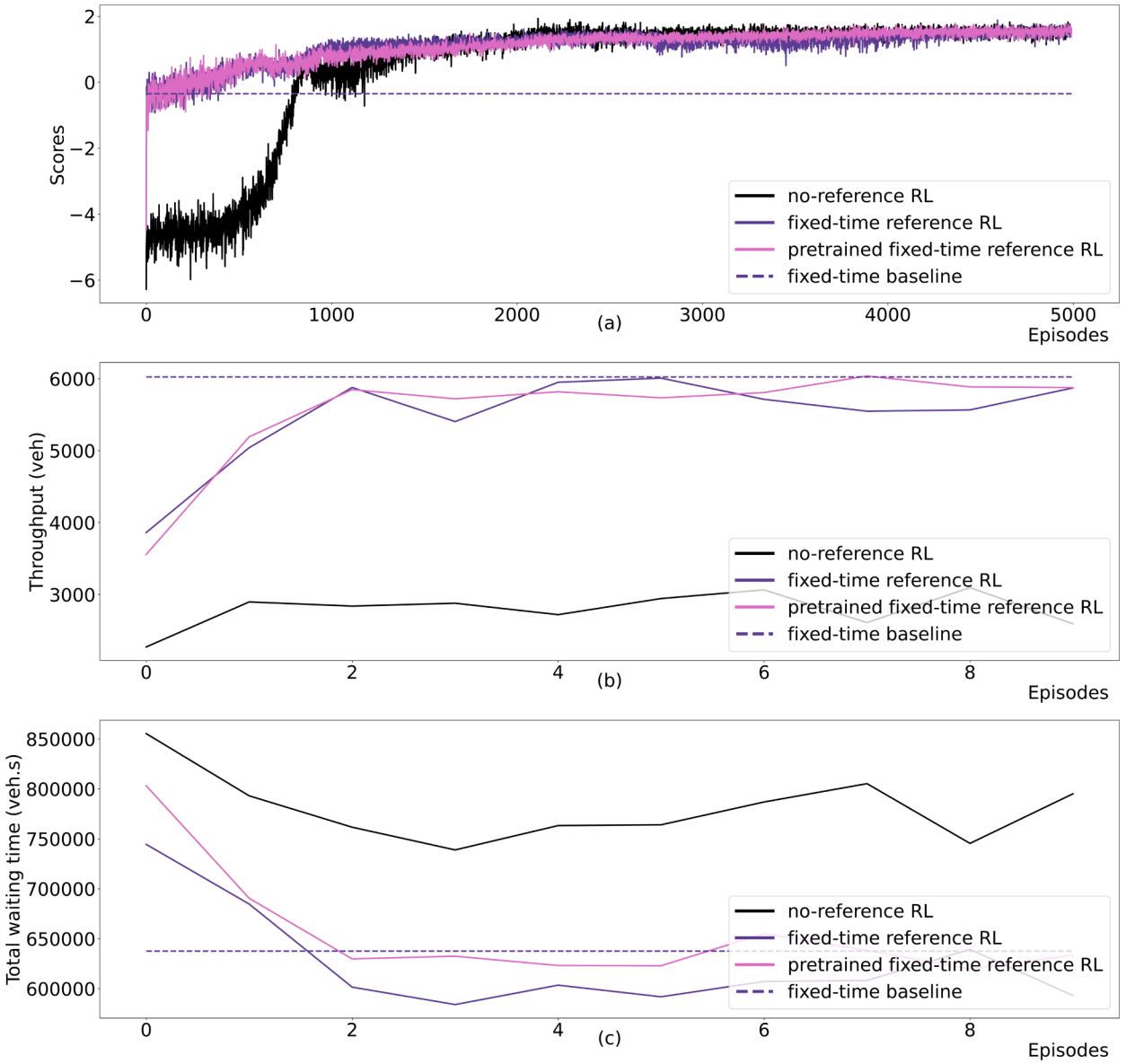

Fig. 15. The learning curves of pretrained fixed-time reference RL. (a) The score progress during training process; (b) The network throughput in the first 10 episodes; (c) The network total waiting time in the first 10 episodes.

scenario.

As depicted in Fig. 11 (a), all three reference RL algorithms demonstrate obvious enhancement in the initial learning performance of the agent. Consistent with the experiment results in the corridor scenario, the BackPressure reference RL achieves the highest initial performance. The adaptive Webster's policy shows similar initial performance to the fixed-time policy. Viewing the entire training process presented in Fig. 11 (b), the fixed-time reference RL attains its baseline after 20 training episodes, improving training speed by 97.7 % compared to the no-reference RL. Meanwhile, the adaptive Webster's reference RL surpasses its baseline performance after 400 training episodes, showcasing a speed enhancement of around 60 % over the no-reference RL. Meanwhile, the BackPressure reference RL achieves its baseline performance after about 1000 training episodes, demonstrating a 50 % acceleration compared to the no-reference RL. Regarding convergence performance, all three reference RL algorithms exhibit higher scores than the no-reference RL.

Figs. 12-14 present the progress of throughput and total waiting time for the real-world network during the training process. As depicted in Figs. 12-14, all reference RL algorithms outperform their corresponding JSRL algorithms after approximately 20 training episodes, and continue to exhibit superior performance during the subsequent training process. Table 3 presents the average results of evaluation metrics during the first 20 training episodes and their convergence values. While JSRL methods initially perform better due to their curriculum design, the reference RL methods outperform them after about 20 training episodes. The superiority of JSRL

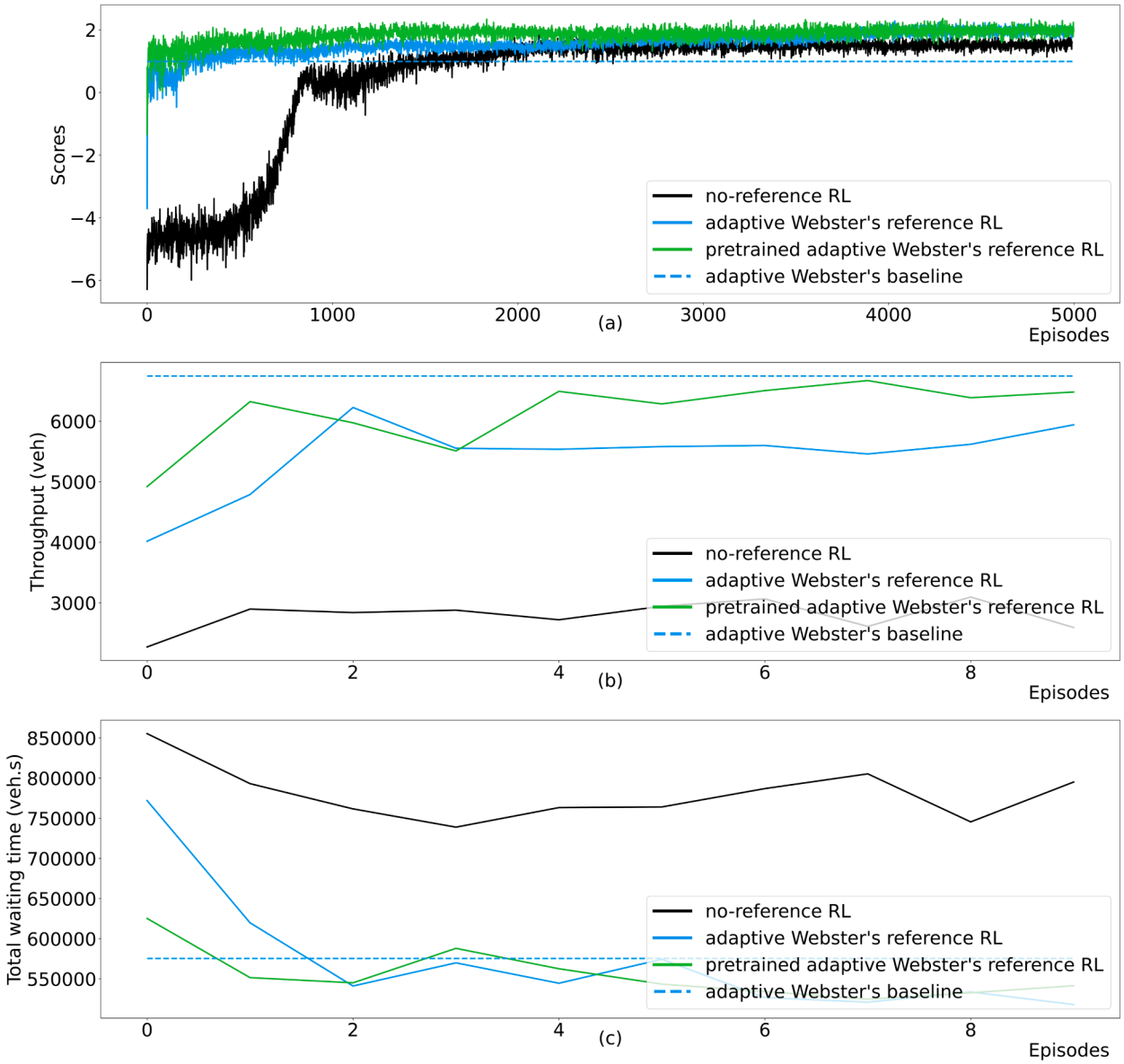

Fig. 16. The learning curves of pretrained adaptive Webster's reference RL. (a) The score progress during training process; (b)The network throughput during the first 10 episodes; (c) The network total waiting time during the first 10 episodes.

algorithms in the early training is attributed to the curriculum design, where nearly all decisions during the first 20 training episodes are made by the guide policy. As control is gradually transferred to the agent, the performance of the JSRL algorithms initially deteriorates and then gradually improve. In terms of convergence performance, the reference RL algorithms achieve comparable network throughput and total waiting time to the JSRL algorithms.

In summary, the proposed reference RL algorithms demonstrate superior performance over their corresponding JSRL algorithms in the real-world network scenario. While JSRL algorithms initially excel due to their curriculum design, the reference RL algorithms surpass them after approximately 20 training episodes. Despite minor differences in early performance, both sets of algorithms achieve comparable convergence in network throughput and total waiting time.

6.4. Performance of the reference RL extensions

6.4.1. The pretrained reference RL

With the pretrained Q-value network (Q_θ), the reference mechanism is able to provide more accurate value estimation early in training process. The Figs. 15-17 plot the learning curves of evaluated methods. In addition, Table 4 provides detailed numerical

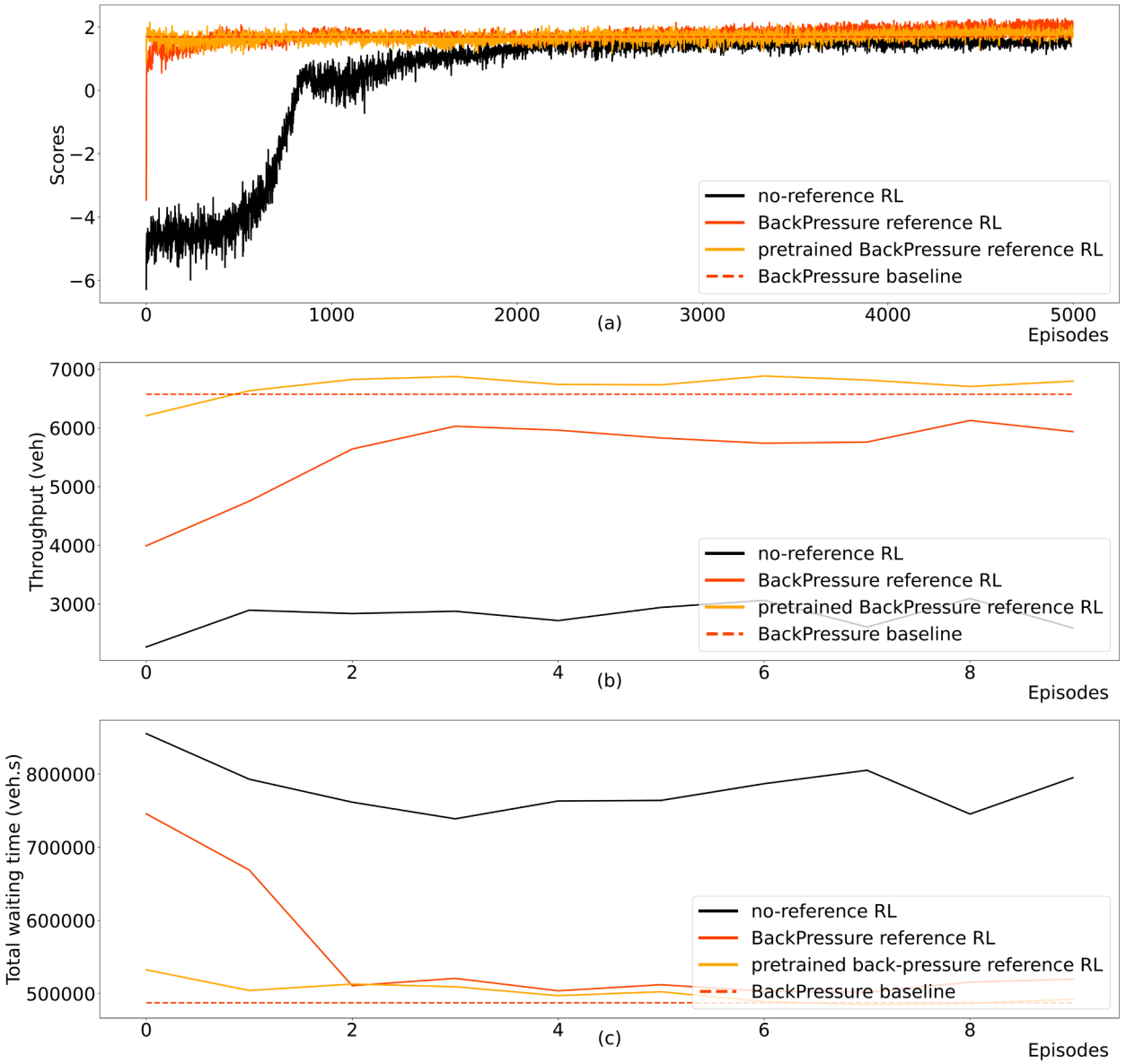

Fig. 17. The learning curves of pretrained BackPressure reference RL. (a) The score progress during training process; (b)The network throughput during the first 10 episodes; (c) The network total waiting time during the first 10 episodes.

Table 4
Performance of the reference RL methods and the pretrained reference RL methods during first 10 episodes.

	fixed-time reference RL	pretrained fixed-time reference RL	Adaptive Webster's reference RL	pretrained adaptive Webster's reference RL	BackPressure reference RL	pretrained BackPressure reference RL
Network throughput (veh)	5484	**5548**	5432	**6156**	5576	**6721**
Total waiting time(s)	625,743	**654,810**	572,060	**554,830**	550,231	**501,126**

results of the first 10 episodes, facilitating an assessment of the performance boost of the pretraining procedure on the initial training episodes.

As shown in Fig. 15, the fixed-time reference RL benefits little from the pretraining process. This may come down to the significant

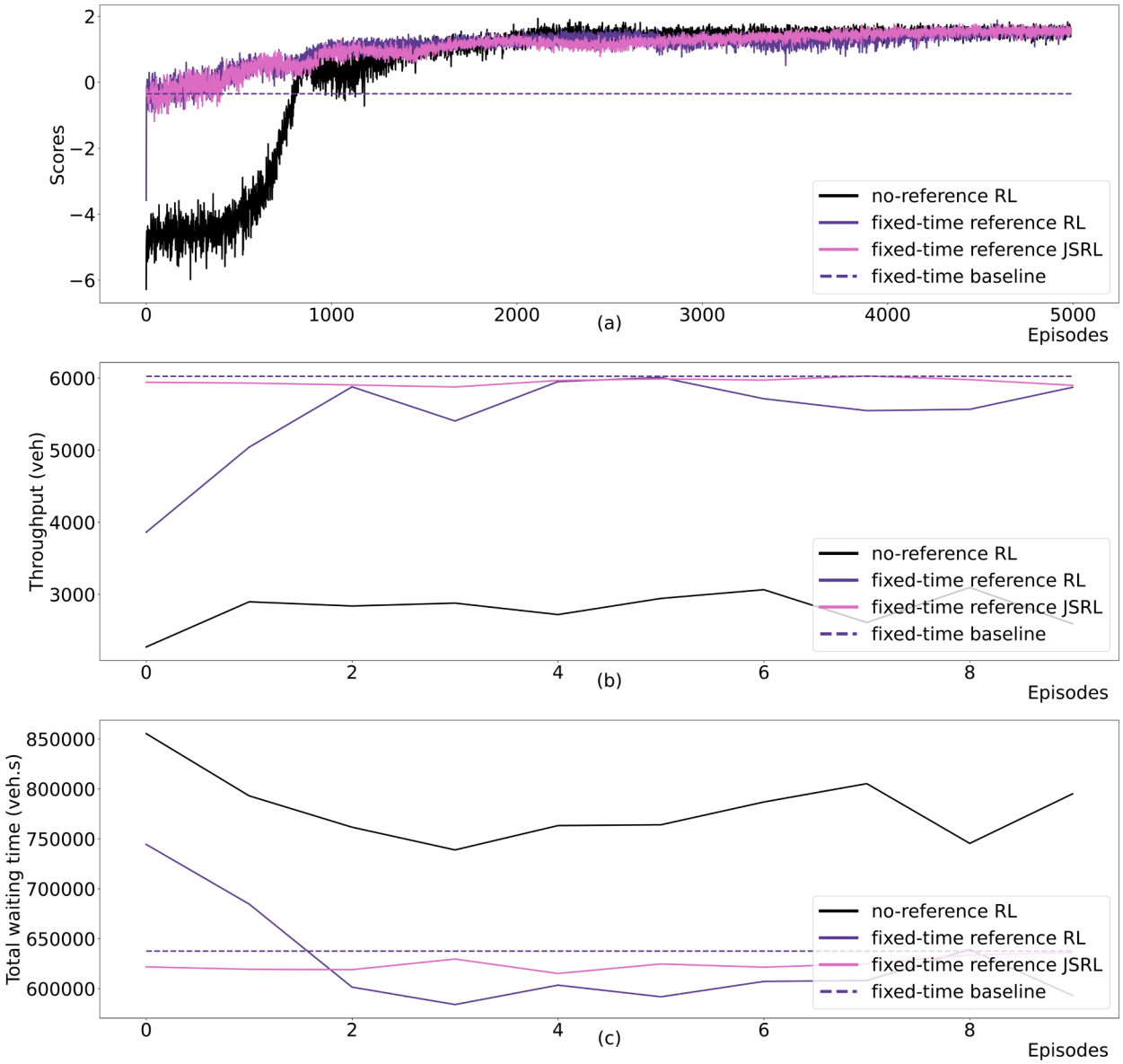

Fig. 18. The learning curves of fixed-reference JSRL. (a) The score progress during training process; (b) The network throughput during the first 10 episodes; (c) The network total waiting time during the first 10 episodes.

shift of action distribution between the pretraining and online training process, as only one signal plan is executed during pretraining. Based on the plots in Figs. 15-17, the pretraining procedure effectively enhance the initial learning performance of the adaptive Webster's reference RL (a 13.3 % increase in network throughput and 3 % reduction in vehicle waiting time) and the BachPressure reference RL (a 20.5 % rise in network throughput and an 8.9 % drop in vehicle waiting time). Additionally, pretraining the Q networks before the online training process does not appear to impact the convergence performance of the agent.

The results reveal that the pretraining procedure effectively improves system performance at the beginning of training process when adopting the adaptive Webster's policy and the BackPressure policy as the agent's reference strategy. Therefore, it is recommended to incorporate an offline pretraining procedure before real-world training, provided the experience data of the reference policy is available or can be easily collected.

6.4.2. The reference JSRL

The reference JSRL leverages the jump-start concept in [4] to the reference-RL framework, aimed at alleviating the inaccurate Q-value evaluations at the early training episodes. Figs. 18-20 depict the learning curves for the reference JSRL methods and their system performance during the first 10 training episodes. Table 5 lists the corresponding numerical results during first 10 training episodes.

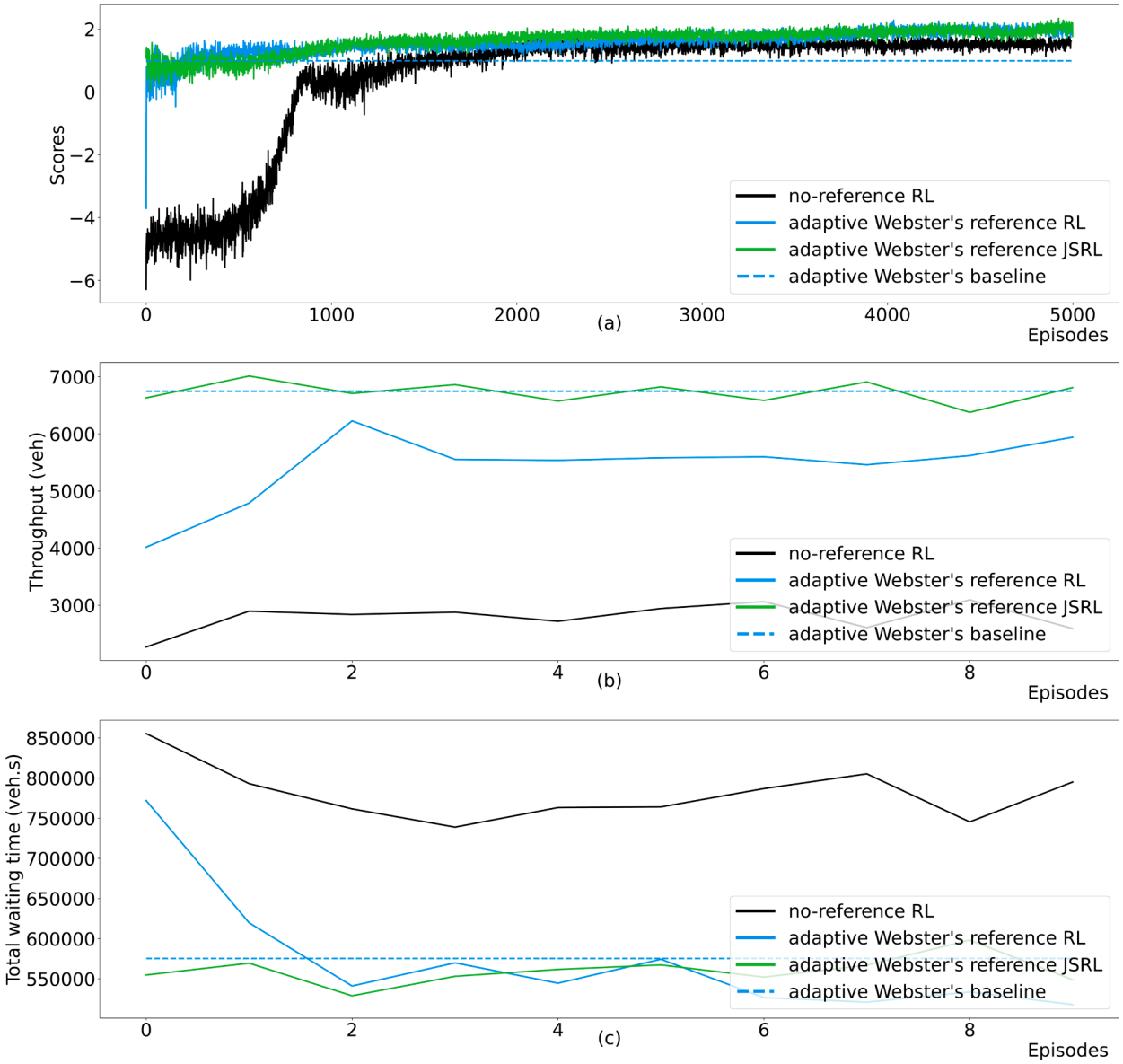

Fig. 19. The learning curves of adaptive Webster's reference JSRL. (a) The score progress during training process; (b) The network throughput during the first 10 episodes; (c) The network total waiting time during the first 10 episodes.

The results in Figs. 18-20 demonstrate that the integration of the reference RL method with the jump-start algorithm yields a notable enhancement in system performance during the early training episodes. Specifically, the fixed-time reference JSRL attains an 8.5 % augmentation in network throughput, and concurrently maintains a vehicle waiting time comparable to that of the fixed-time reference RL approach. The adaptive Webster's reference JSRL exhibits a notable 23.9 % increase in network throughput, coupled with a marginal 2.1 % reduction in total waiting time. The BackPressure reference JSRL approach demonstrates a substantial 17.1 % enhancement in network throughput, alongside a significant reduction of 12.7 % in total waiting time. Meanwhile, the incorporation of the Jump-start concept does not appear to adversely impact convergence performance. Consequently, the integration of the jump-start concept enables the reference RL to maintain a comparable level of performance to its reference policy within the first few training episodes. This is achieved without compromising the learning speed and convergence performance of the agent.

7. Conclusions and future work

Towards training agents in real-world traffic system, the reference RL algorithm is proposed in this paper to improve the system performance during the agent training process, particularly at the early stage of training. By resampling the agent's actions to satisfy

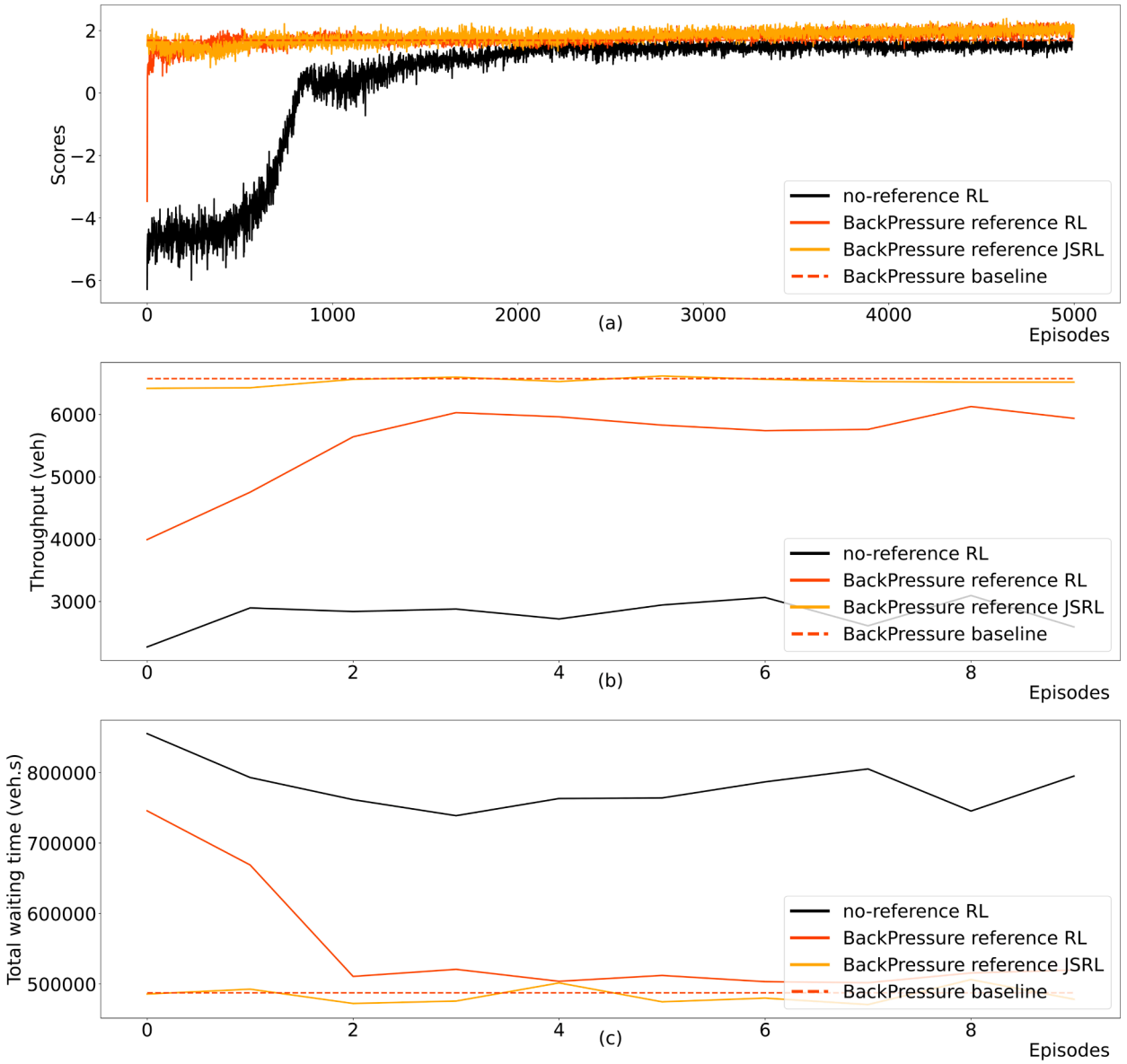

Fig. 20. The learning curves of BackPressure reference JSRL. (a) The score progress during training process; (b) The network throughput during the first 10 episodes; (c)The network total waiting time during the first 10 episodes.

Table 5
Performance of the reference JSRL methods during first 10 episodes.

	fixed-time reference RL	fixed-time reference JSRL	adaptive Webster's reference RL	adaptive Webster's reference JSRL	BackPressure reference RL	BackPressure reference JSRL
Network throughput (veh)	5484	**5948**	5432	**6729**	5576	**6527**
Total waiting time (s)	625,743	**624,552**	572,060	**560,121**	550,231	**483,763**

the Q-value constraints set by the reference policy, the reference RL algorithm accelerates the agent's learning to quickly reach a performance level comparable to its reference policy, thereby speeding up the training process. To further boost the initial learning performance of agent, the pretraining procedure and the jump-start algorithm are combined with the reference RL, respectively. Simulation experiments in both corridor and real-world scenarios demonstrate the superiority of the reference RL, especially in the

initial training stages. Meanwhile, the experiment results suggest that the reference mechanism does not hinder the agent's ability to develop advanced TSC strategies. In addition, the pretraining process can further boost the performance of the reference RL in the first few training episodes. Similarly, the jump-start algorithm helps the reference RL method quickly achieve a performance level similar to its reference policy early in training, without negatively affecting the agent's learning speed and convergence performance.

Several important steps remain for the real-world deployment of the proposed reference RL. Key areas for further investigation include testing scalability in larger and more complex traffic networks, assessing robustness under various traffic environmental conditions, and designing effective communication and cooperation channels between TSC agents. Additionally, this paper focuses on the TSC problem, the border applicability and efficacy of reference RL to other domains necessitate further validation through additional experiments and evaluations. Additional research across various fields is essential to fully understand the potential and limitations of the reference RL in diverse real-world scenarios.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used the ChatGPT in order to improve the readability and language of this article. After using the ChatGPT, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

CRediT authorship contribution statement

Yunxue Lu: Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization. **Andreas Hegyi:** Writing – review & editing, Resources, Conceptualization. **A. Maria Salomons:** Writing – review & editing, Resources, Conceptualization. **Hao Wang:** Supervision, Funding acquisition, Conceptualization.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgements

This work is sponsored by the National Science and Technology Major Project (No. 2022ZD0115600).

Appendix

A. The traffic demand loaded in simulation scenarios

A.1. The corridor network

The intersections in the corridor are labeled as *nt1*, *nt2* and *nt3*, with the peripheral nodes on the west-east roads designated as "*nt0*" and "*nt4*" respectively. The six peripheral nodes of the branch roads are denoted with "*np*" prefixes. Vehicles enter the network with predetermined origin–destination (OD) pairs. In the simulated network, a total of 80 routes are designed, consisting of 40 pairs and their reverse counterparts. The actual routes for vehicles are automatically calculated to select the fastest route when vehicles enter the network. The total traffic volume of movements during a complete simulation episode are illustrated as in Fig. A1.

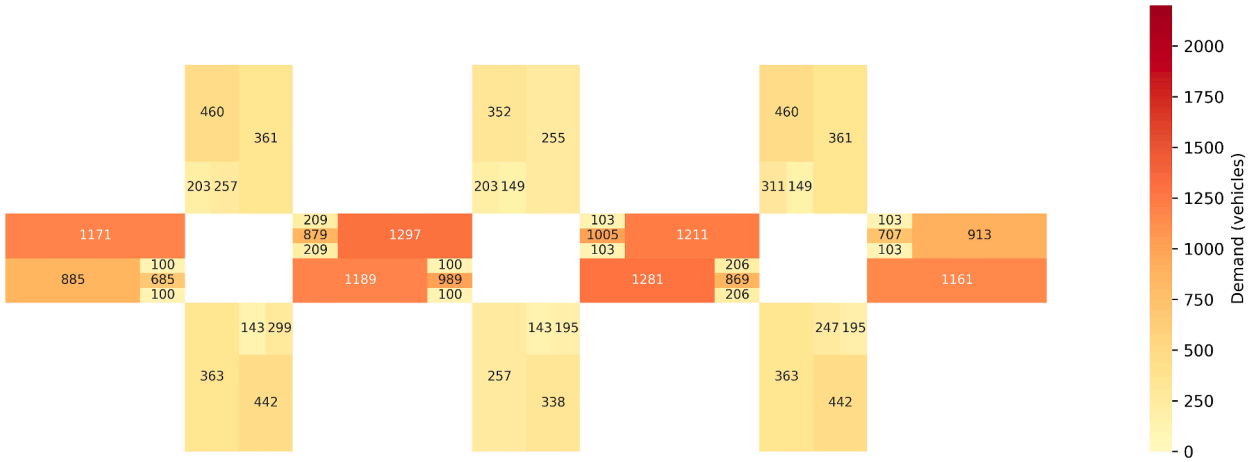

Fig. A1. Traffic demand of intersections.

As shown in Fig. A1, traffic on the west-east roads is much heavier than on the north-east branches. This aligns with typical traffic conditions where the west-east road serves as the primary artery. It is worth noting that the traffic demand is not evenly loaded throughout the episode, but includes obvious peaks to simulate the traffic rushes.

The traffic demand is set at three levels based on OD pair types: $f_{nt-nt} = 400(veh/h)$, $f_{nt-np} = 200(veh/h)$ and $f_{np-np} = 200(veh/h)$. Additionally, in order to simulate traffic rushes, the entry time of vehicles into the network exhibits two distinct peaks. Fig. A2 illustrates the demand dynamics, represented by the ratio of the actual loaded vehicles to peak demand. In Fig. A2, F_1 and F_2 displays the demand loading dynamics for 40 routes in opposite directions.

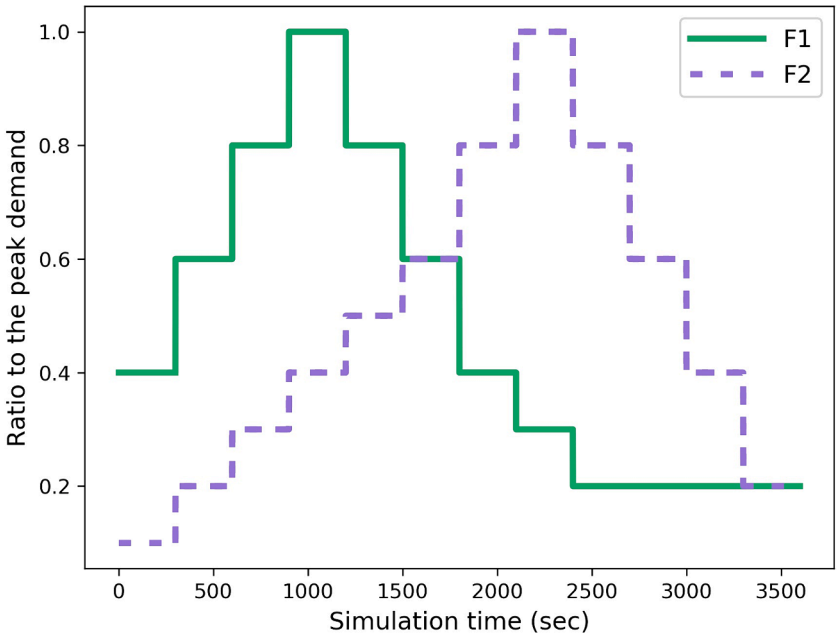

Fig. A2. The fluctuation of traffic demand.

A.2. The traffic demand of the real-world network

The traffic volumes at intersections in the real-world network are shown in the Fig. A3. Similar to the corridor network, vehicles enter the network with predetermined OD road segments, and their actual routes are automatically calculated based on the fastest available path when vehicles entering the network.

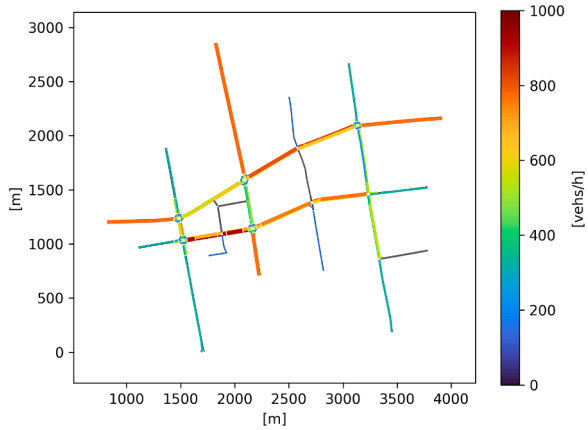

Fig. A3. The traffic demand of real-world network.

In addition, as with the corridor scenario, traffic rushes are simulated in real traffic environment (refer to the Fig. A2).

B. *The hyperparameters and DNN settings of the reference-RL*

Table B1 lists the hyperparameters and the DNN settings for the reference RL method, the JSRL and the MA2C. Since JSRL also employs the SAC algorithm for policy optimization, it shares similar hyperparameter settings with the reference RL.

Table B1
The hyperparameters settings.

Hyper-parameters of reference-RL and JSRL	Value
DNN optimizer	Adam
DNN initializer	Xavier
Learning rate ($\lambda_\sigma, \lambda_\omega, \lambda_\alpha, \lambda_\theta$)	0.0001
Discount (γ)	0.9
Replay buffer size (D^{rl}, D^{ref})	2.10^5
Hidden units per layer in value functions (Q_ω, Q_θ)	[64, 96, 64, 32, 16, 8, 4, 1]
Hidden units per layer in actor network	[64, 96, 64, 32, 16, 8, 4, 2]
Number of samples per minibatch	16
Entropy target ($\overline{H}$)	-1
Nonlinearity	elu
Smoothing coefficient (τ)	0.005
Update interval	3
Updates per learning session	3
Gradient clipping norm	5
Maximum resample times (η)	10
Number of training episodes	5000
Duration of each episode	3600 s
Weights in the reward function (β_1, β_2)	-0.01, 1
Regulation factor (∂)	0.001
Hyperparameters of MA2C	
Batch size	60
Learning rate for actor and critic	5e-4
Reward discount rate	0.99
Control interval	5 s
Yellow duration	2 s
Episode horizon	3600 s
Weight coefficient of reward function	(-1, -0.2)

C. *The state space exploration of the reference-RL*

Digging deeper into the early training stages, Fig. C1 illustrates the agent's explorations in the state space in no-reference RL and the fixed-time reference RL during the first 10,000 learning updates.

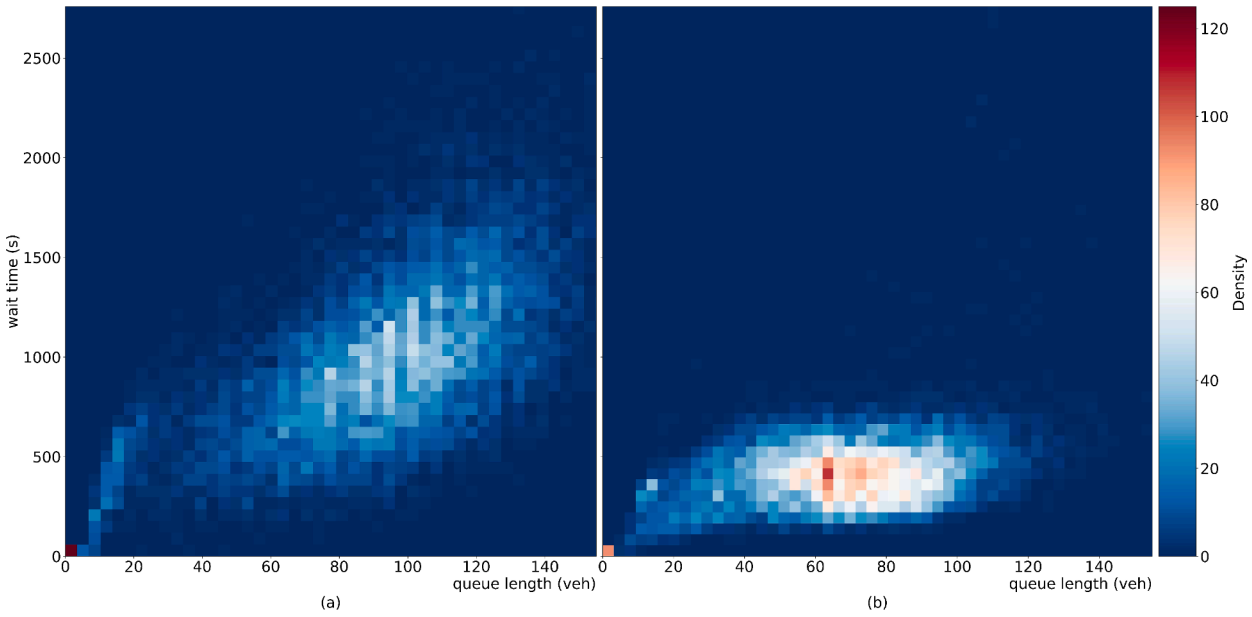

Fig. C1. The exploration of agent in state space. (a) Agent explores without reference mechanism; (b) Agent explores with fixed-time reference policy.

The Fig. C1 illustrates the state space exploration of the agent controlling intersection *nt2*, as shown in Fig. 1 of ***Section 4.1***. The horizontal axis represents the total number of queuing vehicles in both incoming and outgoing lanes of the intersection *nt2*, while the vertical axis indicates the total waiting time of vehicles. The "density" in the color bar reflects the number of data points in a specific range of values, corresponding to the color in the bar. The experimental findings align with our expectations. The agent learning without a reference policy explores a border state space, often encountering states with long queues and extended waiting times. In contrast, the fixed-time reference RL guide the agent toward more favorable state spaces early on, while still allow it to develop proficient control strategies.

Thus, the reference policy can guide the agent away from undesired state spaces, without compromising its ability to learn high-performance control strategies.

References

[1] M. Noaeen, A. Naik, L. Goodman, J. Crebo, T. Abrar, Z.S.H. Abad, A.L.C. Bazzan, B. Far, Reinforcement learning in urban network traffic signal control: a systematic literature review, Expert Syst. Appl. (2022) 116830.
[2] B.-L. Ye, W. Wu, K. Ruan, L. Li, T. Chen, H. Gao, Y. Chen, A survey of model predictive control methods for traffic signal control, IEEE/CAA J. Automatica Sinica 6 (2019) 623–640.
[3] L. Zheng, X. Xue, C. Xu, B. Ran, A stochastic simulation-based optimization method for equitable and efficient network-wide signal timing under uncertainties, Transp. Res. B Methodol. 122 (2019) 287–308.
[4] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao, Jump-Start Reinforcement Learning, ArXiv Preprint ArXiv:2204.02372 (2022).
[5] H. Wei, G. Zheng, V. Gayah, Z. Li, Recent advances in reinforcement learning for traffic signal control: a survey of models and evaluation, (n.d.).
[6] L.A. Prashanth, S. Bhatnagar, Reinforcement learning with function approximation for traffic signal control, IEEE Trans. Intell. Transp. Syst. 12 (2010) 412–421.
[7] L.N. Alegre, T. Ziemke, A.L.C. Bazzan, Using reinforcement learning to control traffic signals in a real-world scenario: an approach based on linear function approximation, IEEE Trans. Intell. Transp. Syst. 23 (2021) 9126–9135.
[8] Z. Li, H. Yu, G. Zhang, S. Dong, C.-Z. Xu, Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning, Transp Res Part C Emerg Technol 125 (2021) 103059.
[9] B. Xu, Y. Wang, Z. Wang, H. Jia, Z. Lu, Hierarchically and cooperatively learning traffic signal control, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021: pp. 669–677.
[10] N. Xiao, L. Yu, J. Yu, P. Chen, Y. Liu, A cold-start-free reinforcement learning approach for traffic signal control, J. Intell. Transp. Syst. 26 (2022) 476–485.
[11] L. Zhang, J. Deng, Data might be enough: bridge real-world traffic signal control using offline reinforcement learning, ArXiv Preprint ArXiv:2303.10828 (2023).
[12] L. Zhu, P. Peng, Z. Lu, X. Wang, Y. Tian, Meta variationally intrinsic motivated reinforcement learning for decentralized traffic signal control, ArXiv Preprint ArXiv:2101.00746 (2021).
[13] E. Liang, Z. Su, C. Fang, R. Zhong, OAM: an option-action reinforcement learning framework for universal multi-intersection control, Proceed. AAAI Conf. Artif. Intell. (2022) 4550–4558.
[14] Y. Zhang, Y. Zhang, R. Su, Pedestrian-safety-aware traffic light control strategy for urban traffic congestion alleviation, IEEE Trans. Intell. Transp. Syst. 22 (2019) 178–193.
[15] Y. Zhang, J. Fricker, Investigating smart traffic signal controllers at signalized crosswalks: A reinforcement learning approach, in: 2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), IEEE, 2021: pp. 1–6.
[16] M. Rodriguez, H. Fathy, Vehicle and traffic light control through gradient-based coordination and control barrier function safety regulation, J. Dyn. Syst. Meas. Contr. 144 (2022).

[17] B. Yu, J. Guo, Q. Zhao, J. Li, W. Rao, Smarter and safer traffic signal controlling via deep reinforcement learning, In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020: pp. 3345–3348.
[18] M. Xu, X. Zhai, Z. Sun, X. Zhou, Y. Chen, Multiagent control approach with multiple traffic signal priority and coordination, J. Transp. Eng. A Syst. 149 (2023) 04022124.
[19] H. Su, Y.D. Zhong, J.Y.J. Chow, B. Dey, L. Jin, EMVLight: a multi-agent reinforcement learning framework for an emergency vehicle decentralized routing and traffic signal control system, Transp. Res. Part C Emerg. Technol. 146 (2023) 103955.
[20] I. Kostrikov, A. Nair, S. Levine, Offline reinforcement learning with implicit q-learning, ArXiv Preprint ArXiv:2110.06169 (2021).
[21] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari, D. Kalashnikov, AW-opt: Learning robotic skills with imitation andreinforcement at scale, in: Conference on Robot Learning, PMLR, 2022, pp. 1078–1088.
[22] S. Lee, Y. Seo, K. Lee, P. Abbeel, J. Shin, Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble, in: Conference on Robot Learning, PMLR, 2022, pp. 1702–1712.
[23] H. Liu, P. Abbeel, Aps: Active pretraining with successor features, in: International Conference on Machine Learning, PMLR, 2021, pp. 6736–6747.
[24] P.-A. Kamienny, J. Tarbouriech, A. Lazaric, L. Denoyer, Direct then diffuse: incremental unsupervised skill discovery for state covering and goal reaching, ArXiv Preprint ArXiv:2110.14457 (2021).
[25] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, D. Pathak, Planning to explore via self-supervised world models, in: International Conference on Machine Learning, PMLR, 2020, pp. 8583–8592.
[26] K. Hakhamaneshi, R. Zhao, A. Zhan, P. Abbeel, M. Laskin, Hierarchical few-shot imitation with skill transition models, ArXiv Preprint ArXiv:2107.08981 (2021).
[27] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, K. Lee, State entropy maximization with random encoders for efficient exploration, in: International Conference on Machine Learning, PMLR, 2021, pp. 9443–9454.
[28] D. Yarats, R. Fergus, A. Lazaric, L. Pinto, Reinforcement learning with prototypical representations, in: International Conference on Machine Learning, PMLR, 2021, pp. 11920–11931.
[29] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, R.D. Hjelm, P. Bachman, A.C. Courville, Pretraining representations for data-efficient reinforcement learning, Adv. Neural Inf. Process Syst. 34 (2021) 12686–12699.
[30] Z. Xie, Z. Lin, J. Li, S. Li, D. Ye, Pretraining in deep reinforcement learning: a survey, ArXiv Preprint ArXiv:2211.03959 (2022).
[31] A. Nair, M. Dalal, A. Gupta, S. Levine, Accelerating online reinforcement learning with offline datasets, ArXiv Preprint ArXiv:2006.09359 (2020).
[32] S. Levine, A. Kumar, G. Tucker, J. Fu, Offline reinforcement learning: tutorial, review, and perspectives on open problems, ArXiv Preprint ArXiv:2005.01643 (2020).
[33] J. Garcıa, F. Fernández, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (2015) 1437–1480.
[34] F.L. da Silva, P. Hernandez-Leal, B. Kartal, M.E. Taylor, Uncertainty-aware action advising for deep reinforcement learning agents, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 5792–5799.
[35] E. Ilhan, J. Gow, D. Perez, Student-initiated action advising via advice novelty, IEEE Trans Games (2021).
[36] E. Ilhan, J. Gow, D. Perez-Liebana, Learning on a Budget via Teacher Imitation, in: 2021 IEEE Conference on Games (CoG), IEEE, 2021, pp. 1–8.
[37] Y. Fu, C. Li, F.R. Yu, T.H. Luan, Y. Zhang, Hybrid autonomous driving guidance strategy combining deep reinforcement learning and expert system, IEEE Trans. Intell. Transp. Syst. (2021).
[38] P.A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, E. Wießner, Microscopic traffic simulation using sumo, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2018, pp. 2575–2582.
[39] A.G. Ptv, Ptv Vissim 7 User Manual, Karlsruhe, Germany, 2015.
[40] M. Lapan, Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, AlphaGo Zero and more, Packt Publishing Ltd, TRPO, 2018.
[41] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, Soft actor-critic algorithms and applications, ArXiv Preprint ArXiv:1812.05905 (2018).
[42] T. Chu, J. Wang, L. Codecà, Z. Li, Multi-agent deep reinforcement learning for large-scale traffic signal control, IEEE Trans. Intell. Transp. Syst. 21 (2019) 1086–1095.
[43] Y. Liu, L. Liu, W.-P. Chen, Intelligent traffic light control using distributed multi-agent Q learning, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2017, pp. 1–8.
[44] J. Liu, H. Zhang, Z. Fu, Y. Wang, Learning scalable multi-agent coordination by spatial differentiation for traffic signal control, Eng. Appl. Artif. Intel. 100 (2021) 104165.
[45] P. Hernandez-Leal, B. Kartal, M.E. Taylor, Is multiagent deep reinforcement learning the answer or the question? A brief survey, Learning 21 (2018) 22.
[46] W. Genders, S. Razavi, An open-source framework for adaptive traffic signal control, ArXiv Preprint ArXiv:1909.00395 (2019).
[47] T. Le, P. Kovács, N. Walton, H.L. Vu, L.L.H. Andrew, S.S.P. Hoogendoorn, Decentralized signal control for urban road networks, Transp Res Part C Emerg Technol 58 (2015) 431–450.
[48] F.V. Webster, Traffic signal settings, 1958.
[49] P. Varaiya, The max-pressure controller for arbitrary networks of signalized intersections, in: Advances in Dynamic Network Modeling in Complex Transportation Systems, Springer, 2013, pp. 27–66.