Mitigating Action Hysteresis in Traffic Signal Control with Traffic Predictive Reinforcement Learning

Xiao Han
hahahenha@gmail.com
City University of Hong Kong

Xiangyu Zhao*
xianzhao@cityu.edu.hk
City University of Hong Kong

Liang Zhang*
zhangliang@sribd.cn
Shenzhen Research Institute of Big Data

Wanyu Wang
wanyuwang4-c@my.cityu.edu.hk
City University of Hong Kong

ABSTRACT

Traffic signal control plays a pivotal role in the management of urban traffic flow. With the rapid advancement of reinforcement learning, the development of signal control methods has seen a significant boost. However, a major challenge in implementing these methods is ensuring that signal lights do not change abruptly, as this can lead to traffic accidents. To mitigate this risk, a time-delay is introduced in the implementation of control actions, but usually has a negative impact on the overall efficacy of the control policy. To address this challenge, this paper presents a novel Traffic Signal Control Framework (PRLight), which leverages an On-policy Traffic Control Model (OTCM) and an Online Traffic Prediction Model (OTPM) to achieve efficient and real-time control of traffic signals. The framework collects multi-source traffic information from a local-view graph in real-time and employs a novel fast attention mechanism to extract relevant traffic features. To be specific, OTCM utilizes the predicted traffic state as input, eliminating the need for communication with other agents and maximizing computational efficiency while ensuring that the most relevant information is used for signal control. The proposed framework was evaluated on both simulated and real-world road networks and compared to various state-of-the-art methods, demonstrating its effectiveness in preventing traffic congestion and accidents.

CCS CONCEPTS

• **Information systems** → *Spatial-temporal systems*; • **Applied computing** → **Transportation**.

KEYWORDS

Traffic Signal Control, Traffic State Prediction, Reinforcement Learning, Graph Convolutional Networks, Attention Mechanism

* Xiangyu Zhao and Liang Zhang are the corresponding authors.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599528

ACM Reference Format:
Xiao Han, Xiangyu Zhao, Liang Zhang, and Wanyu Wang. 2023. Mitigating Action Hysteresis in Traffic Signal Control with Traffic Predictive Reinforcement Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599528

1 INTRODUCTION

Traffic signal control has long been recognized as one of crucial means of mitigating urban traffic congestion, with numerous studies affirming its effectiveness in improving city efficiency and economic growth [9, 27, 35, 46]. Despite this, traditional fixed-timing [20, 30] and hand-crafted methods [5, 23] remain prevalent in the traffic signal control of many cities, which are based on predefined rules and lack the ability to dynamically adjust traffic signals in response to real-time traffic conditions. To address this limitation, there has been a growing interest in the application of novel techniques, such as data-driven reinforcement learning (RL), for traffic signal control. These methods allow learning agents, such as traffic signal controllers, to interact with the environment and make decisions (e.g., selecting the appropriate traffic phase[1]) in real-time [42].

Efforts of RL-based traffic signal control have utilized optimal green time length as the action for controlling traffic lights [13, 14, 19]. These methods calculate the green time length of the next phase when the current action's execution time is about to end. However, a limitation of these approaches is that they do not allow for immediate switching of the traffic light once an action is determined, which may not meet the real-time requirements in practical. For instance, in small to medium-sized cities in China, peak hours can be as short as half an hour, but one signal control cycle in this period can take up to 120-180 seconds. As a result, RL-based methods that only determine optimal actions every cycle may not be able to effectively capture the best traffic control opportunities. On the other hand, other RL-based methods have adopted a timestep-based signal control strategy [28, 37, 45], which allows for real-time switching of traffic lights. These methods have been shown to perform well in real-world road network environments.

In real-world timestep-based traffic control, the implementation of a countdown period, which serves to warn passing vehicles and pedestrians of an upcoming phase change, is a widely accepted

[1] A traffic phase refers to a set of pre-defined traffic signals, displayed for a specific duration, that regulates the movement of vehicles or pedestrians at an intersection.

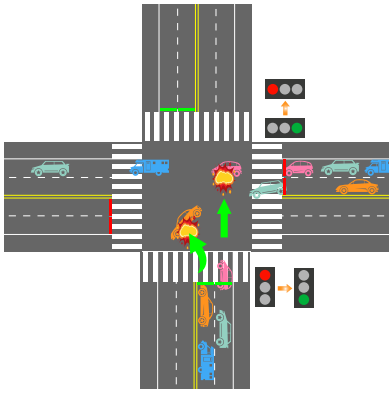


Figure 1: Risk of a sudden change of the signal light.

practice [3]. This is because a sudden change in signal phases increases the likelihood of traffic accidents, as illustrated in Figure 1. To mitigate this risk, solutions have been proposed that extend the amber time or add a brief all-red phase [10, 18, 33]. However, these approaches negatively impact traffic efficiency by reducing the green time of each phase. Thus, the introduction of a countdown period before the switch of signal lights is deemed both reasonable and necessary. However, the issue of action hysteresis in timestep-based traffic control is introduced by the fact that the actual implementation of the phase switch occurs after it is initially decided, which may cause optimal control to be misleading and lead to traffic congestion. Most existing timestep-based signal control methods do not address this problem and its adverse effects.

In this paper, we propose a Traffic Predictive Signal Control Framework (PRLight) to address the problem of action hysteresis in traffic signal control. To achieve this, PRLight incorporates the relationship between current actions and the predicted future traffic state by using a dynamic graph representation method, an Online Traffic Prediction Model (OTPM), and an On-policy Traffic Control Model (OTCM). Specifically, (i) the dynamic graph representation employs a novel graph attention network to extract dynamic graph representations from real-time traffic features with an improved fast attention mechanism based on Taylor expansion. (ii) OTPM provides short-term feature prediction to assist in resolving the issue of action hysteresis based on the graph representations. (iii) OTCM models the traffic signal control over multiple intersections as a multi-agent reinforcement learning (MARL) problem. The predicted traffic state, instead of the observed one, is used as input for the control policy to evaluate the impact of delayed actions better. The policy of each agent in MARL is trained efficiently using a novel distributed framework, where each agent can train its policy independently. The training balances the limitations of local observations and the high computational costs of global observations by using a local-view graph, which includes the most relevant global traffic information. This eliminates the need for communication among agents and maximizes computational efficiency.

In summary, our contributions are demonstrated as follows:

- We propose a framework, PRLight, which uses MARL to model signal control over intersections and incorporates short-term predicted traffic state as input to address the issue of action hysteresis due to countdown. To the best of our knowledge, this is a novel approach that has not been explored previously;
- We develop a novel attention-based graph learning network to extract dynamic graph representations from real-time traffic features, which utilizes an improved fast attention mechanism based on Taylor expansion to accelerate the computation process;
- We present a distributed training algorithm for the policy of agents in MARL, allowing each agent to train independently without sacrificing global views;
- Extensive experiments based on both simulated and real-world road networks demonstrate the effectiveness and efficiency of the framework PRLight.

2 PROBLEM FORMULATION

In this paper, we adopt the MARL framework to solve the problem of real-time traffic signal control. More specifically, we consider a traffic signal control machine at an intersection to be an agent. Each agent can obtain environmental traffic information and control one intersection independently. Before introducing our MARL framework, we first provide one crucial definition.

Definition 1. ***Phase lock/unlock stage.*** *When a signal light enters the countdown stage or $T_{\min}$ time is left in the green light, the current phase is defined as the phase lock stage. Otherwise, it is defined as the phase unlock stage.*

The purpose of the phase lock stage is to prevent the real-time signal light from suddenly changing and ensure that vehicles and pedestrians can safely pass through the intersection. In this stage, the signal light does not accept any additional control actions to avoid traffic accidents. In the phase unlock stage, the signal light could accept any control actions that may maintain or switch the current traffic phase after a countdown.

Then, we demonstrate our MARL framework. We consider a global road graph $\mathcal{G}^g$ with N signal-controlled intersections and related traffic features X_t^g, and utilize N agents to control these intersections. For each agent u, we define five key elements:

- **Decision Time ($[T]$)**: It is a set of all finite decision timesteps $[T] = \{1, \cdots, t, \cdots, T\}$. It includes a sequential process in which the number of vehicles entering the road network gradually increases from 0, and then all leave the road network completely.
- **Action ($\mathcal{A}^u$)**: $\mathcal{A}^u = \{a_0^u, \cdots, a_t^u, \cdots, a_T^u\}$ represents the set of actions actually performed by a traffic light in the decision cycle. $a_t^u := \{0, 1\}$ is an action performed by the signal controller u at time t, where $a_t^u = 0$ is to maintain the current traffic phase at time t, $a_t^u = 1$ means to switch to the phase lock stage and then change to the next phase after the countdown ends.
- **State ($\mathcal{S}^u$)**: $\mathcal{S}^u = \{S_0^u, \cdots, S_t^u, \cdots, S_T^u\}$ represents the set of traffic states observed at each time t. Here S_t^u is an embedding matrix extracted from the dynamic graph $\mathcal{G}^u$ and traffic features X_t^u.
- **Reward ($\mathcal{R}^u$)**: $\mathcal{R}^u = \{r_0^u, \cdots, r_t^u, \cdots, r_T^u\}$ represents the set of rewards calculated by a function of the (state, action, next state) tuple, i.e., $r^u(s, a, s') = \mathbb{E}[r_{t+1}^u | S_t^u = s, a_t^u = a, S_{t+1}^u = s']$, and it is predefined according to the change of queuing length at each lane in an intersection. The total return is defined as $\sum_t \gamma \cdot r_t^u$, where γ is a discount factor, $\gamma \in [0, 1]$.

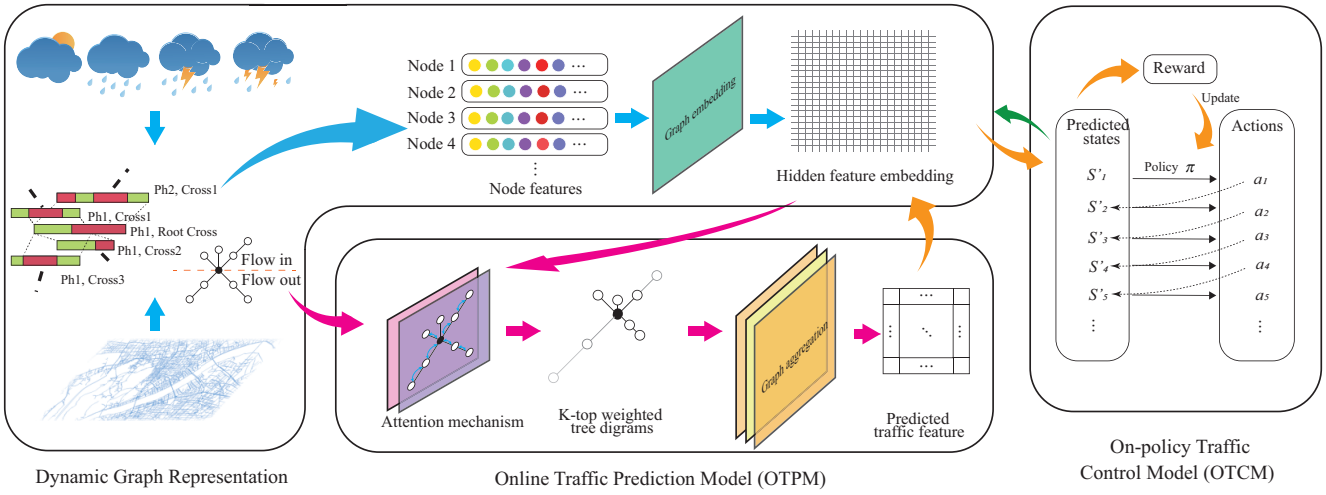


Figure 2: The framework overview of PRLight.

- **Policy (π_θ^u):** $\pi_\theta^u = \pi_\theta^u(a|s)$ is a mapping from states to actions of the u-th agent.

Based on the definitions above, we formulate the problem of the multi-agent traffic signal control system as follows.

Problem 1. *Given the directed road graph $\mathcal{G}^g$ and the observed traffic features $\boldsymbol{X}_t^g$ at time $t \in [T]$, the goal of MARL is to learn a series of policy $\pi_\Theta = \left\{\pi_\theta^u | u = 1, \cdots, N\right\}$ that maximize the total reward of average waiting rate for all intersections in $\mathcal{G}^g$ during timesteps $[T]$.*

In the above problem, all agents share the same global road graph $\mathcal{G}^g$ and traffic features $\boldsymbol{X}_t^g$. In reality, it is time-consuming and unnecessary for each agent to process them independently. In our paper, we use a local road graph $\mathcal{G}^u \subseteq \mathcal{G}^g$ and its traffic features $\boldsymbol{X}_t^u$ to capture the local traffic information related to agent u. Then, agent u can use RL method to interact with the environment with local information and updates the policy π_θ^u over time.

3 METHODOLOGY

In this section, we first provide a framework overview of PRLight. Then, we introduce the dynamic graph representation learning method to capture the local traffic state information of each gent. After that, we introduce our Online Traffic Prediction Model (OTPM), which can assist in policy updating. Next, we demonstrate the On-policy Traffic Control Model (OTCM), which optimizes the policy of each agent. At last, we demonstrate the distributed training process.

3.1 Overall Framework

Figure 2 shows the overall framework, which consists of a dynamic graph representation, an Online Traffic Prediction Model (OTPM), and an On-policy Control Model (OTCM). The first procedure collects static data (e.g., road networks and points of interest) and dynamic data (e.g., the phases of traffic lights and road volume) from the environment. Then, it fuses all data into a local spatiotemporal graph centered on the target intersection with high-dimension features as the graph signal within each short period. To describe this graph in real-time and more accurately, we introduced a fast attention method for modeling. After that, OTPM takes the dynamic traffic spatiotemporal local graph and related local features as input and outputs predicted traffic features for a given time in the future. To reduce the negative impact of delayed signal control actions, OTCM uses the predicted traffic features instead of observed ones as input and outputs the optimal action.

3.2 Dynamic Graph Representation

Following the prior practice [11], we represent the global road network as a directed graph $\mathcal{G}^g$ in which each lane is considered a node v and the upstream or downstream relationship between two lanes is regarded as a directed edge. Formally, we define a global road digraph as $\mathcal{G}^g = \{\boldsymbol{V}^g, \boldsymbol{E}^g\}$, where $\boldsymbol{V}^g = \{v_1, \cdots, v_{|V^g|}\}$ stands for the set of all nodes, $\boldsymbol{E}^g = \{e_{ij} : v_j \text{ is downstream of } v_i\}$ is the set of directed edges. We use $\boldsymbol{X}_t^g \in \mathbb{R}^{|\boldsymbol{V}^g| \times m}$ as the m-dimensional traffic features at time t, where each dimension represents one traffic flow parameter, e.g., traffic flow and traffic speed. Lanes in a path with the same path direction are treated as different nodes with the same road traffic features. To better understand the process of road graph representation, we demonstrate a road network with 3 intersections and a total of 13 lanes shown in Figure 3(a) and the corresponding road digraph with 13 nodes shown in Figure 3(b).

The digraph size of $\mathcal{G}^g$ is usually large and hard to be processed on time. To reduce the size of the road graph, we focus on a local view of the graph rooted by a set of nodes $\boldsymbol{V}^u$ in a specific intersection u. In this local-view graph, we set up a constant $n \leq |\boldsymbol{V}^g|$ to limit the maximum node number. We consider the lanes nearer to intersection u with the higher priority of joining the set $\boldsymbol{V}^u$ until $|\boldsymbol{V}^u| = n$. This method captures the most important lanes to the traffic state in intersection u. Then, we can obtain the local-view graph $\mathcal{G}^u = \{\boldsymbol{V}^u, \boldsymbol{E}^u\}$ according to $\mathcal{G}^g$ and the traffic features $\boldsymbol{X}_t^u$ according to $\boldsymbol{X}_t^g$. An example of a local-view graph with $n = 6$ is shown in Figure 4 for the road network shown in Figure 3(a).

With little abuse of notations, we do not demonstrate the index of variables referring to a specific agent or intersection u if without

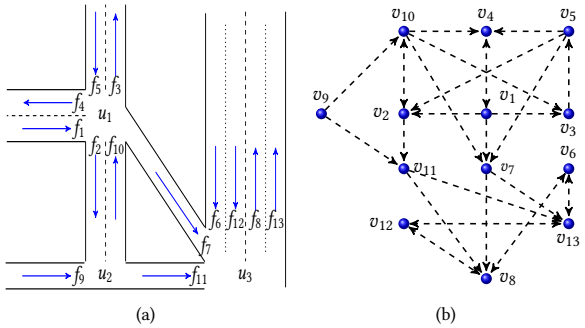


Figure 3: A sample of digraph representation.

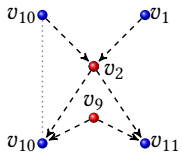


Figure 4: An example of local-view graph: This graph has a tree-like structure centered at intersection u_2, with two entry lanes, f_2 and f_9, forming the root node set $\{v_2, v_9\}$. When setting $n = 6$, 4 additional nodes are added including upstream lanes f_{10}, f_1 and downstream lanes f_{10} and f_{11}. The dotted line is added between nodes that present the same lane.

a specified statement in the rest paper, e.g., we use $\mathcal{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$ to refer $\mathcal{G}^u = \{\boldsymbol{V}^u, \boldsymbol{E}^u\}$ for agent u by default. Note that two connected lanes from v_i to v_j in road graph $\mathcal{G}$ do not indicate that the vehicles can move through them at any time. For instance, when the traffic light is in the red phase, the vehicles in lane v_i are obstructed and cannot move to v_j. To capture the scenarios, we define a dynamic graph as $\mathcal{G}_t = \{\boldsymbol{V}, \boldsymbol{E}_t\}$ with the same node $\boldsymbol{V}$, but dynamic edges $\boldsymbol{E}_t$. Obviously, the edge $e_{ij} \in \boldsymbol{E}$ is not connected when the traffic light is in the red phase and connected when the traffic light is in the green phase. When the traffic light is in a countdown stage, some vehicles may hesitate to move through the intersection or even stop. In this case, we define this edge as partially connected. Formally, we define the dynamic weight of edges in $\mathcal{G}_t$ as follows:

$$\boldsymbol{A}_t^{(ij)} = \begin{cases} 1, & \text{if } t_\mathrm{r} \geq t_\mathrm{cd}, \\ t_\mathrm{r}/t_\mathrm{cd}, & \text{if } t_\mathrm{r} < t_\mathrm{cd}, \\ 0, & \text{if } e_{ij} \text{ is in the red phase or } i = j, \end{cases} \tag{1}$$

where $\boldsymbol{A}_t^{(ij)}$ is the weight of edge $e_{ij} \in \boldsymbol{E}$ at time t, and when $\boldsymbol{A}_t^{(ij)} = 0$, we consider the link between nodes $v_i, v_j \in V$ to be cut off; t_r is the remaining green time in the current phase, which is related to the current time t; t_cd is countdown time.

The weights of edges in Equation (1) reflect the road conditions but not the traffic states. The traffic features between two connected edges in $\mathcal{G}$ are also important for building edge relationships between two lanes. For instance, an intersection with serious traffic jams usually takes more time to move through than that without any vehicles. To capture the complex relationships, we use attention mechanism [32] to update the weight of edges in $\mathcal{G}_t$ as follows:

$$\begin{aligned} \boldsymbol{A}'_t &= \mathrm{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\mathrm{T})\boldsymbol{V} \\ &= \left(\frac{\sum_{j=1}^n e^{q_i^\mathrm{T} k_j}}{\sum_{l=1}^n \sum_{j=1}^n e^{q_l^\mathrm{T} k_j}} v_i \right)_{i=1}^n, \end{aligned} \tag{2}$$

where $\boldsymbol{Q} = \mathrm{Linear}_{W_Q}(\boldsymbol{X}_t) \in \mathbb{R}^{n \times d_Q}$, $\boldsymbol{K} = \mathrm{Linear}_{W_K}(\boldsymbol{X}_t) \in \mathbb{R}^{n \times d_Q}$, $\boldsymbol{V} = \mathrm{Linear}_{W_V}(\boldsymbol{A}_t + \boldsymbol{A}_t^\mathrm{T} + \boldsymbol{I}_n) \in \mathbb{R}^{n \times n}$, $d_Q << n$; $\boldsymbol{X}_t$ is the road traffic feature; q_i, k_i, v_i are i-th row vectors of matrices $\boldsymbol{Q}, \boldsymbol{K}$ and $\boldsymbol{V}$.

Note that the attention mechanism applied in Equation (2) has extremely high computational costs. If we can reduce the computational complexity of this step, it would be helpful to improve the real-time performance of constructing a dynamic graph. Observing Equation (2), if we could remove the SoftMax function and multiply $\boldsymbol{K}^\mathrm{T}$ and $\boldsymbol{V}$ at first, then the computational cost could be reduced. Thus, we propose a *fast attention* method shown in Theorem 1 to substitute the $Att(\cdot)$ function in Equation (2).

Theorem 1. *Using the Taylor expansion and a set of different training weights W'_Q and W'_K, we could rewrite Equation* (2) *as:*

$$Att(\boldsymbol{Q}', \boldsymbol{K}', \boldsymbol{V}) \approx \boldsymbol{Q}'(\boldsymbol{K}'^\mathrm{T}\boldsymbol{V}) - (\boldsymbol{Q}' \odot \boldsymbol{Q}') \left[(\boldsymbol{K}'^\mathrm{T} \odot \boldsymbol{K}'^\mathrm{T})\boldsymbol{V} \right], \tag{3}$$

where $\boldsymbol{Q}' = \mathrm{Linear}_{W'_Q}(\boldsymbol{X})$, $\boldsymbol{K}' = \mathrm{Linear}_{W'_K}(\boldsymbol{X})$, and $\odot$ is the Hadamard product.

Proof. The proof can be found in Appendix B.1. Besides, the numerical experiments are done in Section 4.5 to show the efficiency of the derived formula in the theorem, and the settings of these experiments are introduced in Appendix B.2 in detail. □

Finally, a graph convolutional layer (GCL) with ReLU activation function σ and a dropout layer, as shown in Equation (4), is used to learn the representation of the dynamic graph at time t.

$$\boldsymbol{H}_t = \sigma\left(\boldsymbol{D}'^{-\frac{1}{2}}_t \boldsymbol{A}'_t \boldsymbol{D}'^{-\frac{1}{2}}_t \boldsymbol{X}_t \cdot W_\mathrm{GCL} \right), \tag{4}$$

where $\boldsymbol{D}'_t = \mathrm{diag}(d'_{ii})_{i=1}^n$ is the degree matrix at timestamp t, $d'_{ii} = \sum_j \boldsymbol{A}'^{(ij)}$, W_GCL is the training weight. We summarize the parameters in this representation as $\mathbf{W}_G = \left\{ W'_Q, W'_K, W_V, W_\mathrm{GCL} \right\}$. Note that parameters $\mathbf{W}_G$ in this subsection are shared by all agents.

3.3 Online Traffic Prediction Model

As we mentioned previously, real-world traffic control usually designs a countdown stage to prevent the real-time traffic light from suddenly changing so as to avoid traffic accidents. The countdown stage can result in a countdown delay t_cd for the effectiveness of actual action control, which might mislead the optimization of light control. For instance, an upstream lane may maintain the green light to allow vehicles to pass to the downstream lane if the downstream is observed without any traffic jam. However, the traffic jam might turn out after $t_\mathrm{cd}/2$ due to the traffic control at the next intersection. The vehicles in the upstream lane would still enter the downstream lane after $t_\mathrm{cd}/2$ time due to the countdown stage delay. To solve this delay, we propose an online traffic prediction model (OTPM) to predict the traffic after t_cd time. The prediction can assist in traffic control to estimate its impacts properly [29].

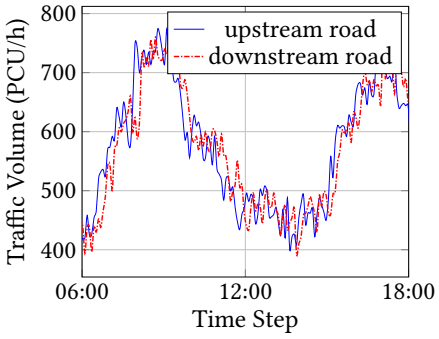


Figure 5: Trends of traffic volume of adjacency roads: When the traffic volume on the upstream road increases/decreases, the traffic volume on the downstream road will also change accordingly after a short period of time.

Traffic prediction is not an easy task. Firstly, there exist multiple factors that affect the traffic state like weather, traffic jams, etc. Secondly, due to traffic control, the traffic state of the current lane can be changed dynamically. To solve these issues, we analyze the traffic pattern between adjacent roads. As shown in Figure 5, we find that their traffic patterns are highly correlated. This provides the foundations of traffic predictions. On the other hand, we can also observe that there exists an around 10 minutes traffic delay between these roads. It indicates that the impact of traffic control for the upstream lane would also have 10 minutes delay. Fortunately, the countdown time t_{cd} is usually much smaller than such delay. This indicates that we can ignore the impacts of traffic control for other intersections when we predict the traffic only t_{cd} time later.

In OTPM, we use the dynamic graph representation in the previous subsection to extract the complex traffic relationships among different intersections. Then, we further use a Multi-Layer Perceptron (MLP) to predict short-term traffic features, shown as follows:

$$\boldsymbol{X}'_{t+t_{\text{cd}}} = \text{MLP}_{W_{\text{p}}}(\boldsymbol{H}_t) = \sigma\left(\text{Linear}_{W_{\text{p}}}(\boldsymbol{H}_t)\right), \tag{5}$$

where $\sigma(\cdot)$ is the ReLU function and W_{p} is the weight matrix.

To optimize the parameters in OPTM, we measure the distance between the predicted traffic features and the ground truth. Besides, we use regularization to avoid overfitting. Formally, we have:

$$\arg\min_{\mathbf{W}_P} L_P(\mathbf{W}_P) = D(\boldsymbol{X}'_{t+t_{\text{cd}}}, \boldsymbol{X}_{t+t_{\text{cd}}}; \mathbf{W}_P) + |\mathbf{W}_P|_2 + ||\mathbf{W}_P||_2^2, \tag{6}$$

where $\mathbf{W}_P = \{\mathbf{W}_G, W_{\text{p}}\}$ and $D(\cdot,\cdot)$ uses Euclidean Distance to measure the Root of Mean Square Error (RMSE) loss.

3.4 On-policy Traffic Control Model

As previously mentioned, we use the MARL with N agents to learn the light control in the N signal-controlled intersections. We use the clip-Proximal Policy Optimization (clip-PPO) [26] framework to learn the optimal policy for all agents in MARL. Then, the optimization objective can be expressed as follows:

$$\max_{\{\theta^1,\ldots,\theta^N\}} \mathbb{E}_{\boldsymbol{\pi}_{\text{old}}}\left\{\sum_{u=1}^{N} f\left(Pr_t\left(\theta^u\right), A^{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{S}_t^g, \boldsymbol{a}_t)\right)\right\}, \tag{7}$$

where $\boldsymbol{a}_t = \{a_t^{(1)}, \cdots, a_t^{(N)}\}$ is the joint action of all agent at time t, $\boldsymbol{S}_t^g$ is the global state embedding extracted from road graph $\mathcal{G}^g$ and traffic features X_t^g, and

$$\begin{aligned} &f\left(Pr_t\left(\theta^u\right), A^{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{S}_t^g, \boldsymbol{a}_t)\right) \\ &= \min\left\{Pr_t\left(\theta^u\right) A^{\pi_{\text{old}}}(\boldsymbol{S}_t^g, \boldsymbol{a}_t), c^{\varepsilon}\left(Pr_t\left(\theta^u\right)\right) A^{\pi_{\text{old}}}(\boldsymbol{S}_t^g, \boldsymbol{a}_t)\right\}. \end{aligned} \tag{8}$$

In Equation (8), the clip function $c^{\varepsilon}(x)$ restricts x into the interval $[1-\varepsilon, 1+\varepsilon]$, $Pr_t\left(\theta^u\right) = \frac{\pi^u(a_t^u|S_t^u)}{\pi_{old}^u(a_t^u|S_t^u)}$ is the probability ratio for agent u, and $A^{\pi_{\text{old}}}(\boldsymbol{S}_t^g, \boldsymbol{a}_t)$ is estimated advantage function.

Learning the optimization objective in Equation (7) directly is usually time-consuming. Recently, the MARL algorithm like IPPO [6] and MAPPO [43] learn the objective for each agent separately. Each agent in IPPO uses local observations to learn the value function, which limits the ability to evaluate the global state. The MAPPO adopts all global observations to estimate the value function. However, it may suffer from large amounts of computations due to the large graph size of $\mathcal{G}^g$. In our OTCM, we use the global information from local-view graph $\mathcal{G}^u$, which extracts the most important information related to agent u and alleviates the burden of computations. In addition, we use the future state $\boldsymbol{S'}^u_{t+t_{\text{cd}}}$ from predicted features $\boldsymbol{X'}^u_{t+t_{\text{cd}}}$ to learn the policy instead of the current state $\boldsymbol{S}_t^u$ to solve the countdown delay problem. Formally, the optimization objective in OTCM for agent u can be expressed:

$$\max_{\theta^u} \mathbb{E}_{\pi^u_{\text{old}}}\left\{f\left(\frac{\pi^u\left(a_t^u \mid \boldsymbol{S'}^u_{t+t_{\text{cd}}}\right)}{\pi^u_{\text{old}}\left(a_t^u \mid \boldsymbol{S'}^u_{t+t_{\text{cd}}}\right)}, A^{\pi^u_{\text{old}}}(\boldsymbol{S}_t^u)\right)\right\}. \tag{9}$$

To extract the state $\boldsymbol{S'}^u_{t+t_{\text{cd}}}$ and $\boldsymbol{S}_t^u$ in Equation (9), we use the dynamic graph representation module in Section 3.2. We denote the dynamic graph representation for $\boldsymbol{X'}^u_{t+t_{\text{cd}}}$ as $\boldsymbol{H'}^u_{t+t_{\text{cd}}}$ and that for $\boldsymbol{X}_t^u$ as $\boldsymbol{H}_t^u$. Then, we let $\boldsymbol{S'}^u_{t+t_{\text{cd}}} = \boldsymbol{H'}^u_{t+t_{\text{cd}}}$ and $\boldsymbol{S}_t^u = \boldsymbol{H}_t^u$.

Based on the state, we use an MLP for the u-th agent to further calculate the probability of each different action:

$$\pi_{\theta^u}(a_t^u|\boldsymbol{S'}^u_{t+t_{\text{cd}}}) = \text{MLP}_{W_a^u}(\boldsymbol{S'}^u_{t+t_{\text{cd}}}), \tag{10}$$

where $\theta^u = \{\mathbf{W}_G, W_a^u\}$ is the parameters of policy for agent u.

The advantage function in Equation (9) is estimated by the truncated version of generalized advantage estimation (GAE) in [25],

$$A_t^u = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^T\delta_T, \tag{11}$$

where $\delta_t = r_t + \gamma V_{\omega^u}\left(\boldsymbol{S}_{t+1}^u\right) - V_{\omega^u}\left(\boldsymbol{S}_t^u\right)$, V_{ω^u} is the target local critic of agent u, r_t is the reward, and an MLP is used to define the critic network:

$$V_{\omega^u}(\boldsymbol{S}_t^u) = MLP_{W_c^u}(\boldsymbol{S}_t^u), \tag{12}$$

where $\omega^u = \{\mathbf{W}_G, W_c^u\}$ is parameter set of it.

The critic function V also needs to be updated to approach the accurate value function. According to [7], we update each critic by minimizing the loss function:

$$\min_{\omega^u} \mathbb{E}_t\left[\left(r_t^u + \gamma V_{\bar{\omega}^u}\left(\boldsymbol{S}_{t+1}^u\right) - V_{\omega^u}\left(\boldsymbol{S}_t^u\right)\right)^2\right], \tag{13}$$

where $V_{\bar{\omega}^u}$ is the target state-value function and the parameters $\bar{\omega}^u$ are periodically updated with the most recent ω^u to stabilize learning process.

Algorithm 1 PRLight Training Process

Input: Real-time observation $\boldsymbol{X}_t$ and local-view graph $\mathcal{G}^u$,
Output: parameter $\mathbf{W}_P$ for OTPM and θ^u, ω^u for OTCM.

Global OTPM:
1: Initialize and deliver the global parameters $\mathbf{W}_P$
2: **for** Each timestep $t = 1, \cdots$ **do**
3: **if** At least one local OTPM uploads paramters **then**
4: Update: $\mathbf{W}_P = \frac{1}{Z} \sum_{u=u_1,\cdots,u_Z} \mathbf{W}_P^u$
5: Deliver the global parameter $\mathbf{W}_P$
6: **end if**
7: **end for**

Local Agent u**:**
8: Initialize parameters $\tilde{\theta}^u$, $\tilde{\omega}^u$, θ^u, ω^u and two buffers $\mathbf{B}_P^u$, $\mathbf{B}_C^u$
9: **for** Each timestep $t = t_{\text{cd}}, \cdots$ **do**
10: Observe the real-time traffic feature: $\boldsymbol{X}_t^u$
11: Calculate dynamic weight $\mathbf{A}_t^u$ and graph embedding $\boldsymbol{H}_t^u$
12: Update buffer $\mathbf{B}_P^u$ by sample $\{(\boldsymbol{X}_{t-t_{cd}}^u, \mathbf{A}_t^u, \boldsymbol{X}_t^u)\}$

Local OTPM:
13: **if** An update from global model **then**
14: $\mathbf{W}_P^u \leftarrow \mathbf{W}_P$
15: **end if**
16: Update $\mathbf{W}_P^u \leftarrow$ **Training**$(\mathbf{B}_P^u)$
17: Predict $\boldsymbol{X'}_{t+t_{\text{cd}}}^u$ based on $\boldsymbol{H}_t^u$

OTCM:
18: Compute the future state embedding: $\boldsymbol{S'}_{t+t_{\text{cd}}}^u$
19: Sample an control action a_t^u by policy $\pi_{\theta^u}(a_t^u | \boldsymbol{S'}_{t+t_{\text{cd}}}^u)$
20: Observe the next-step traffic features $\boldsymbol{X}_{t+1}^u$
21: Compute the reward: r_t^u
22: Update buffer $\mathbf{B}_C^u$ by sample $\{(\boldsymbol{S'}_{t+t_{\text{cd}}}^u, a_t^u, r_t^u)\}$
23: **if** Length$(\mathbf{B}_C^u)$ exceeds the limit **then**
24: Delete the oldest element
25: **end if**
26: Estimate the Advantage: A_t^u
27: Update actor network: $\tilde{\theta}^u$, θ^u
28: Update critic network: $\tilde{\omega}^u$, ω^u
29: Upload the parameter $\mathbf{W}_P^u$
30: **end for**
31: **return** trained parameters $\mathbf{W}^u$, θ^u and ω^u

3.5 Training Process

In this subsection, we introduce the training process of the PRLight framework, shown in Algorithm 1. We divide the training process into two types of servers, i.e., a global server that aims to synchronize the global parameters in OTPM and N local servers which provide distributed training process for agents. The global server updates the global parameters $\mathbf{W}_P$ by calculating the average of uploaded local parameters $\mathbf{W}_P^u$ and delivers the updated global parameter to the local server, shown in lines 1-7.

The local server for each agent maintains two buffers, i.e., $\mathbf{B}_P^u$ collecting data for training parameters in local OTPM and $\mathbf{B}_C^u$ collecting samples for training the policy in OTCM. When a new traffic feature $\mathbf{X}_t^u$ is observed, the buffer $\mathbf{B}_P^u$ is updated at first, shown in lines 10-12. After updating the local parameters $\mathbf{W}_P^u$ in OTCM to the latest from the global server, the agent trains the traffic prediction model and updates the local parameters $\mathbf{W}_P^u$, shown in lines 13-16.

Table 1: Statistics of road network datasets.

	Sim	XS
Total intersections	26	80
Signal-controlled intersections	4	27
Total lanes	192	504
Records of link relations	226	983
Coverage (km^2)	5	10

The local OTPM then provides a prediction for the traffic feature at t_{cd} steps later, shown in line 17. With the traffic feature prediction, the local server updates the parameters in OTCM. The agent calculates the feature state at time $t + t_{\text{cd}}$ according to the predicted traffic feature, shown in line 18. It then samples a control action and obtains the next-step traffic features and reward, shown in lines 19-21. The buffer $\mathbf{B}_C^u$ is updated with the new sample and always keeps the latest ones, shown in lines 22-25. Based on samples in $\mathbf{B}_C^u$, the agent updates the parameters in actor by maximizing Equation (9) and that in critic by minimizing Equation (13), shown in lines 26-28. Finally, the local parameters $\mathbf{W}_P^u$ are uploaded to the global server, shown in line 29. Note that $\mathbf{W}_P^u$ is also updated when we update the parameters of actor θ^u and that of critic ω^u.

4 EXPERIMENTS

This section conducts extensive experiments using various road network datasets to evaluate the effectiveness of PRLight. We first introduce the experimental settings, then compare PRLight with representative baselines, and then conduct ablation studies and efficiency analysis. Finally, a case study is introduced.

4.1 Dataset

We use two traffic road networks for experiments: one is a simulated road network (Sim), and the other is a real-world road network (XS). The details of these datasets are listed in Table 1.

We define 4 types of vehicles running on the network: electric bikes, cars, trucks, and buses. These vehicles have different driving behaviors w.r.t. maximum speed, acceleration, etc. We collect average speed, traffic volume, and queue length as the features of lanes. Note that only the signal-controlled intersection can be controlled and optimized by an agent. We simulate a total of 26 intersections in the Sim road network, 4 of which can be controlled by traffic lights. As for the XS road network dataset, we use the real-wold road network from Xiaoshan, a city in China, which consists of 80 intersections and 27 signal-controlled intersections. We collect historical data within one month, which include 3 static features (length, speed limit, road level) of each lane and 11 dynamic features (average speed, occupancy rate, traffic volume, queue length, temp, pasta, visibility, precipitation, rel-humidity, wind direction, and wind velocity) generated every 5-6 seconds. More details about the datasets can be found in Appendix C.1.

4.2 Experimental Settings

Baselines We compare PRLight with baselines from both conventional and RL-based signal control methods, including Fixed [20], VA [39], DynSTGAT [38], FRAP [47], Colight [34], HiLight [40],

Table 2: Experimental results of different baselines.

	Waiting Rate (%)		Travel Time Loss	
	Sim	XS	Sim	XS
Fixed	73.5941	36.1728	0.8551	0.5717
VA	77.3659	47.2597	0.8734	0.6772
DynSTGAT	60.1411	35.4862	0.6987	0.4989
UniLight	71.6804	44.2943	0.8092	0.5962
Colight	67.3384	38.3403	0.7328	0.5442
FRAP	65.2313	37.1411	0.7127	0.5287
HiLight	57.9878	33.1588	0.6256	0.4591
MaCAR	58.4120	32.4520	0.6594	0.4607
PRLight	**55.3044***	**31.4885***	**0.6049***	**0.4492***

"*" indicates the statistically significant improvements
(i.e., two-sided t-test with p < 0.05) over the best baseline.
For all metrics: the lower, the better.

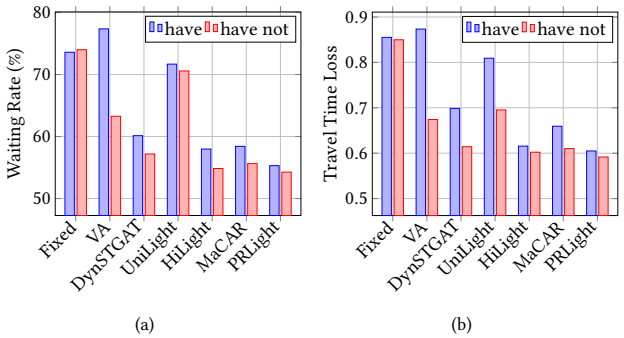


Figure 6: Experimental result of different control settings.

UniLight [15] and Macar [44]. More details about these baselines are shown in Appendix C.2. For ablation studies, we compare the variants of PRLight to verify the effectiveness of each component.

Evaluation metrics Since each vehicle has a different driving trajectory and destination, we prefer not to use *average travel time* directly in this work. Instead, we use the metrics *waiting rate* and *travel time loss* of the whole road network, following the studies in the field of traffic signal control [41]. The definitions of the two metrics are demonstrated as follows:

- **Waiting Rate:** It is the average percentage ratio of each vehicle's waiting time to its total travel time;
- **Travel Time Loss:** It is the average ratio of the total number of seconds a vehicle lost due to traveling slower than expected to this vehicle's total travel time.

Implementation Details We have three modules in the proposed framework: graph representation method, OTPM, and OTCM. In the graph representation method, we set $d_Q = 100$, and the hidden dimension of the graph convolutional layer is 128. In addition, we set the node number of a local graph $n_{\text{Sim}} = 192$ for the Sim dataset and $n_{\text{XS}} = 500$ for the XS dataset. The countdown time is set as $t_{cd} = 10s$ by default. In OTPM, the output feature dimension is the same as the feature dimension of X_t^u. The size 1000 is set for the length of all replay buffers. As for OTCM, we set all hidden dimensions in different layers to be 128, and the model coefficients $\gamma = 0.99, \varepsilon = 0.2$. We train the buffered sample 10 times between every two timestamps. The rate $dp = 0.5$ is set for all dropout layers in PRLight. In addition, Adam is used as the optimizer for all models, and we use the default parameter for it, i.e., $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $lr = 10^{-3}$.

4.3 Overall Performance

The performance of all the baselines in two datasets is shown in Table 2, in terms of the two metrics, i.e., waiting rate and travel time loss. The performance of all methods is the average of the last 5 runs in a total of 25 runs. We can see that the conventional methods have poor performance in the simulated dataset, i.e., Fixed and VA. This is because traffic can be different and dynamic over time, and these methods rely on predefined rules heavily. Once traffic conditions change unexpectedly, these rules may all fail. However, for the real-world data, the fixed time control method outperforms some state-of-the-art reinforcement learning-based methods, e.g., UniLight. The reason may be that signal timing staff set the fixed rules according to some domain knowledge.

DynSTGAT, UniLight, FRAP, Colight, HiLight, and MaCAR are six methods using reinforcement learning as a part of their methods. Compared to these methods, we find that PRLight still performs the best of them in both datasets. This is because most of them cannot fully achieve predictive control of traffic lights, especially when there is a 10-second countdown before the phase-change action is applied. Among the baselines, only MaCAR can predict traffic flow data in units of signal cycles. Note that one signal period of some lights can exceed 150 seconds during peak hours, and some even have 180 seconds. Compared with MaCAR, our prediction model only needs to predict the traffic state of the road network after 10 seconds, and the accuracy of our prediction model is around 88.3% while MaCAR is only about 85.3% under the same settings, which means that our model is easier to be trained.

4.4 Ablation Study

The influence of countdown As mentioned previously, the countdown stage can affect the design of the signal light control. To demonstrate the impact of a countdown stage on control methods, we conduct experiments on different models in the Sim dataset. Specifically, we consider two cases: one is a traffic light with a 10s countdown, and the other is without any countdown time. The experimental results can be shown in Figure 6.

Figure 6(a) shows the waiting rate of models and Figure 6(b) shows the travel time loss of models. We can observe that the existence of a countdown has little effect on the fixed-time method since it always changes phases with a predefined green light time cycle by cycle, which is also consistent with people's general impression. Besides, the countdown has the greatest impact on the VA method, and the delayed signal control instructions will greatly increase the number of vehicles queuing in the road network and increase the time for vehicles to stop and wait at an intersection. We can also find that the effect of a countdown on our framework, PRLight, is second only to the fixed timing method, which is mainly due to the fact that our scheme adopts an independent prediction model, and the training process of the prediction model is also parallel to the training of the traffic control model.

Table 3: Experimental results with/without prediction model.

	Waiting Rate		Travel Time Loss	
	Sim	XS	Sim	XS
with OTPM	55.3044	31.4885	0.6049	0.4492
without OTPM	58.2601	33.1007	0.6278	0.4601

The effect of OTPM To better understand the role of OTPM in our model. We mask the output of OTPM and only use the real-time traffic state as the input of OTCM. The results of comparison experiments can be shown in Table 3. We can observe that the framework's performance drops by around 5.5% for the waiting rate and around 4% for travel time loss. Without OTPM, the performance of PRLight is similar to the performance of HiLight and MaCAR, as shown in Table 2. OTPM is a key module for aggregating surrounding road network information and an indispensable part of interaction with the surrounding environment in PRLight. Therefore, PRLight without the OTPM model will be inferior to HiLight when comparing Table 2 and Table 3.

The effect of RL methods Note that our OTCM provides a general MARL framework, which trains each agent independently. We adopts Clip-PPO for each agent in OTCM. Yet, other RL methods can be also adopted. To show the effect of different RL methods, we compare our model by replacing Clip-PPO with DQN [21], REINFORCE [36], Double Deep Q-Learning (DDQN) [31] and Actor-Critic [2] models in the Sim dataset. The result is shown in Figure 7.

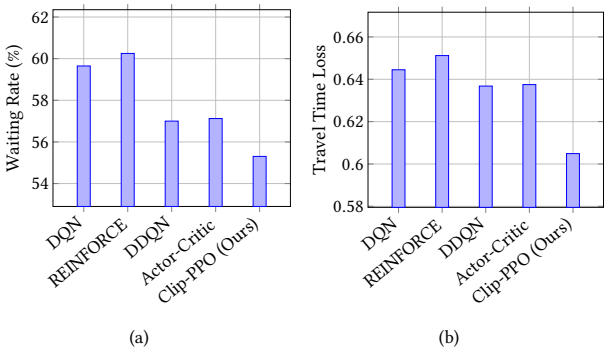


Figure 7: Comparision of different RL models.

As shown in Figure 7, Clip-PPO performs the best in our MARL framework for OTCM. DQN only learns an action-value function, and DDQN separates the two operations of selecting an action and calculating the value of an action on the basis of DQN. Both methods only learn one action-value function. Clip-PPO extends the policy-based methods like the Actor-Critic algorithm by limiting the change of policy, which guarantees training stability. We also observe that the choice of RL methods has a great impact on the performance, but our MARL framework plays a more important role. For example, our MARL framework with simple DQN has performed better than two RL baselines, DynSTGAT and UniLight.

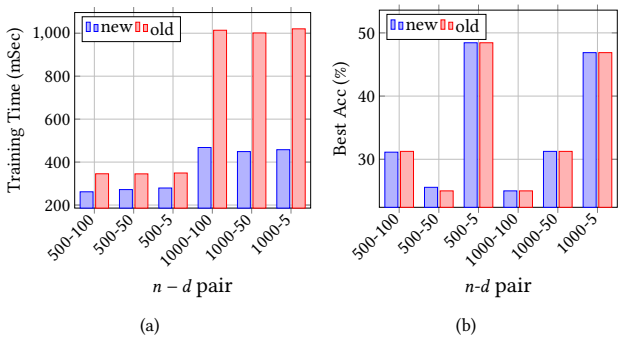


Figure 8: Comparison of different attention methods.

4.5 Efficiency Study

Attention efficiency We propose a fast attention method to reduce the computational cost. To verify the efficiency, we conduct experiments on this module alone via setting the number of nodes $n = 500$ and 1000, the number of feature dimensions $d = 5, 50, 100$. More detailed settings and results can be found in Appendix B.2.

In Figure 8, our proposed method shows a great improvement over the original attention mechanism-based method. It can be observed in Figure 8(a) that when n gets larger, the fast attention module takes up less average training time for every epoch. Besides, when $n >> d$, the value of d has a negligible effect on the results of this experiment. On the other hand, Figure 8(b) demonstrates that even though the fast attention method is an approximating method, the accuracy is almost the same as the original one.

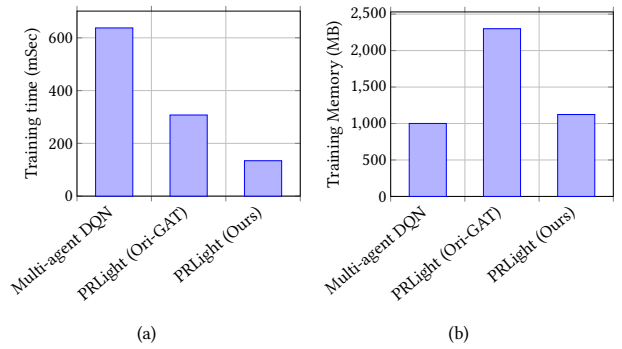


Figure 9: Training time of different models for one epoch.

Training efficiency We test the training efficiency of our PRLight framework in XS dataset. We use the time cost for training samples in one epoch as a metric to evaluate it. We compare our framework with a multi-agent DQN method and PRLight with the original attention mechanism, shown in Figure 9. It shows that PRLight takes the least time to train samples in one epoch. As for the training memory, our method can effectively reduce high memory usage caused by the attention mechanism, shown in 9(b).

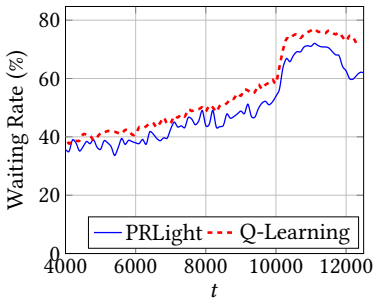


Figure 10: Trends in waiting rates over time.

(a) Q-Learning based control (b) Signal control with PRLight

Figure 11: Visualization of traffic state at $t = 12000$.

4.6 Case Study

In the case study, we take a local area of the XS road network. Since the real-world traffic in this area is smooth most of the time, we use simulated traffic data. We continuously add vehicles to the road network with a fixed probability while ensuring that the ratio of vehicles exiting over vehicles entering the road network satisfies a Gaussian distribution. We expect to observe a change in the average waiting rate of vehicles from a smooth state to a congested state.

Figure 10 demonstrates the overall waiting rate as the number of vehicles increases with time. Our method can effectively reduce the average waiting rate of vehicles when compared with Q-learning methods. When the number of vehicles becomes larger, this phenomenon becomes more obvious. Figure 11 provides a spatial visualization of the traffic state in the local area of the XS road network at peak time $t = 12000$. The red color represents the roads that are congested, while the green color represents the roads that are unimpeded. It shows the control result of PRLight is better than the Q-learning-based control method. When compared with the Q-learning-based method shown in Figure 11(a), although there still exist congested roads in PRLight shown in Figure 11(b), the probability of road congestion is significantly reduced, and the overall traffic efficiency of the road network is improved. This is mainly because our model can predict the short-term future traffic states from a global perspective so as to assist in the control in advance.

5 RELATED WORK

In this section, we briefly review related traffic signal control research including traditional methods and RL-based methods.

Traditional signal control methods are mainly divided into three methods: fixed-time [20], induction control [12], and adaptive control [39]. The fixed-time method sets a fixed green light time for each phase to give all directions of traffic an equal opportunity to enter an intersection. Inductive control [12] detects the number of passing vehicles and then adjusts the green light time based on it. Adaptive signal control [39] uses man-made rules and real-time traffic data to optimize traffic lights intelligently to meet various control goals. The main disadvantage of these methods is that either they cannot effectively adjust the traffic light according to road conditions or they require a lot of human intervention.

The development of RL technology provides a new direction to solve the above problems. Depending on the modeling philosophy of [1, 16], it divides current RL methods into two categories: model-based methods and model-free methods. As a model-based method, Cahill et al. [24] propose a decentralized RL-based urban traffic control optimization scheme that includes a nonparametric pattern change detection mechanism to identify local traffic patterns. Khamis et al. [17] develop a multiagent traffic light control system based on a multi-objective sequential decision-making framework. The model-based methods require the transition probability of the traffic environment, which is usually difficult to obtain due to the complex traffic trends and people's driving behavior. As for model-free methods, Nishi et al. [22] develop an RL-based traffic signal control method that employs a graph convolutional neural network. Chu et al. [4] develop a fully scalable and decentralized MARL algorithm for the advantage actor-critic (A2C) within the context of adaptive traffic signal control. Wei et al. [35] propose a more effective deep reinforcement learning model for traffic light control based on a large-scale real traffic dataset obtained from surveillance cameras. However, all these model-free methods neglect the impacts of a countdown, which limit their performances.

6 CONCLUSION

In this paper, a novel framework called PRLight is proposed to address the problem of action hysteresis in traffic signal control. Specifically, our framework can be divided into three important modules, i.e., the dynamic graph representation module, OTPM for short-term prediction, and OTCM for signal control. In the dynamic graph representation module, we proposed a novel attention-based graph network to capture the information from dynamic graphs fastly. In OTPM, we provided the short-term prediction for traffic features to assist in solving action hysteresis issues. In OTCM, we proposed a novel MARL framework, which adopts a distributed training approach without scarifying the global view. We would like to improve the generalization of our PRLight to apply in more real-word scenarios with missing and abnormal data in the future.

ACKNOWLEDGMENTS

This research was partially supported by APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of City University of Hong Kong), CityU - HKIDS Early Career Research Grant (No.9360163), SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), Tencent (CCF-Tencent Open Fund), Huawei (Huawei Innovation Research Program) and Ant Group (CCF-Ant Research Fund, Ant Group Research Fund) , InnoHK initiative, the Government of the HKSAR, and the Laboratory for AI-Powered Financial Technologies.

REFERENCES

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[2] Andrew G Barto, Richard S Sutton, and Charles W Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* 5 (1983), 834–846.

[3] Yu-Chiun Chiou and Chien-Hua Chang. 2010. Driver responses to green and red vehicular signal countdown displays: Safety and efficiency aspects. *Accident Analysis & Prevention* 42, 4 (2010), 1057–1065.

[4] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. 2019. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems* 21, 3 (2019), 1086–1095.

[5] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. 2013. Self-organizing traffic lights: A realistic simulation. In *Advances in applied self-organizing systems*. Springer, 45–55.

[6] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. 2020. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533* (2020).

[7] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[8] David L Gerlough and Matthew J Huber. 1976. *Traffic flow theory*. Technical Report.

[9] Qiangqiang Guo, Li Li, and Xuegang (Jeff) Ban. 2019. Urban traffic signal control with connected and automated vehicles: A survey. *Transportation Research Part C: Emerging Technologies* 101 (2019), 313–334. https://doi.org/10.1016/j.trc.2019.01.026

[10] Ke Han and Vikash V Gayah. 2015. Continuum signalized junction model for dynamic traffic networks: Offset, spillback, and multiple signal phases. *Transportation Research Part B: Methodological* 77 (2015), 213–239.

[11] Xiao Han, Guojiang Shen, Xi Yang, and Xiangjie Kong. 2020. Congestion recognition for hybrid urban road systems via digraph convolutional network. *Transportation Research Part C: Emerging Technologies* 121 (2020), 102877.

[12] JR Head. 1982. A new specification for inductive loop detectors. *Traffic Engineering & Control* 23, 4 (1982).

[13] Jingzhi Hu, Hongliang Zhang, Lingyang Song, Zhu Han, and H Vincent Poor. 2020. Reinforcement learning for a cellular internet of UAVs: Protocol design, trajectory control, and resource management. *IEEE Wireless Communications* 27, 1 (2020), 116–123.

[14] Jingzhi Hu, Hongliang Zhang, Lingyang Song, Robert Schober, and H Vincent Poor. 2020. Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning. *IEEE Transactions on Communications* 68, 11 (2020), 6807–6821.

[15] Qize Jiang, Minhao Qin, Shengmin Shi, Weiwei Sun, and Baihua Zheng. 2022. Multi-Agent Reinforcement Learning for Traffic Signal Control through Universal Communication Method. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). 3854–3860.

[16] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[17] Mohamed A Khamis, Walid Gomaa, and Hisham El-Shishiny. 2012. Multi-objective traffic light control system based on Bayesian probability interpretation. In *2012 15th International IEEE conference on intelligent transportation systems*. IEEE, 995–1000.

[18] Neetesh Kumar, Syed Shameerur Rahman, and Navin Dhakad. 2020. Fuzzy inference enabled deep reinforcement learning-based traffic light control for intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems* 22, 8 (2020), 4919–4928.

[19] Xiaoyuan Liang, Xunsheng Du, Guiling Wang, and Zhu Han. 2019. A deep reinforcement learning network for traffic light cycle control. *IEEE Transactions on Vehicular Technology* 68, 2 (2019), 1243–1253.

[20] Alan J. Miller. [n. d.]. Settings for Fixed-Cycle Traffic Signals. 14, 4 ([n. d.]), 373–386. https://doi.org/10.1057/jors.1963.61

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[22] Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura. 2018. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *2018 21st International conference on intelligent transportation systems (ITSC)*. IEEE, 877–883.

[23] Isaac Porche and Stéphane Lafortune. 1999. Adaptive look-ahead optimization of traffic signals. *Journal of Intelligent Transportation System* 4, 3-4 (1999), 209–254.

[24] As' ad Salkham and Vinny Cahill. 2010. Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *13th international IEEE conference on intelligent transportation systems*. IEEE, 531–538.

[25] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[27] Guojiang Shen, Xiao Han, KwaiSang Chin, and Xiangjie Kong. 2021. An attention-based digraph convolution network enabled framework for congestion recognition in three-dimensional road networks. *IEEE Transactions on Intelligent Transportation Systems* 23, 9 (2021), 14413–14426.

[28] Yifan Tang and Yan Xu. 2021. Multi-agent deep reinforcement learning for solving large-scale air traffic flow management problem: A time-step sequential decision approach. In *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. IEEE, 1–10.

[29] Anna Izabel J Tostes, Fátima de LP Duarte-Figueiredo, Renato Assunção, Juliana Salles, and Antonio AF Loureiro. 2013. From data to knowledge: City-wide traffic flows analysis and prediction using bing maps. In *Proceedings of the 2nd ACM SIGKDD international workshop on urban computing*. 1–8.

[30] Elise Van der Pol and Frans A Oliehoek. 2016. Coordinated deep reinforcement learners for traffic light control. *Proceedings of learning, inference and control of multi-agent systems (at NIPS 2016)* 1 (2016).

[31] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[33] Fo Vo Webster. 1958. *Traffic signal settings*. Technical Report.

[34] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. 2019. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1913–1922.

[35] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2496–2505.

[36] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[37] Cathy Wu, Kanaad Parvate, Nishant Kheterpal, Leah Dickstein, Ankur Mehta, Eugene Vinitsky, and Alexandre M Bayen. 2017. Framework for control and deep reinforcement learning in traffic. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–8.

[38] Libing Wu, Min Wang, Dan Wu, and Jia Wu. 2021. DynSTGAT: Dynamic Spatial-Temporal Graph Attention Network for Traffic Signal Control. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2150–2159.

[39] Richard Wunderlich, Cuibi Liu, Itamar Elhanany, and Tom Urbanik. 2008. A novel signal-scheduling algorithm with quality-of-service provisioning for an isolated intersection. *IEEE Transactions on Intelligent Transportation Systems* 9, 3 (2008), 536–547.

[40] Bingyu Xu, Yaowei Wang, Zhaozhi Wang, Huizhu Jia, and Zongqing Lu. 2021. Hierarchically and cooperatively learning traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 669–677.

[41] Y Yamazaki, K Shigematsu, S Kato, F Kojima, H Onari, and S Takata. 2017. Design method of material handling systems for lean automation—Integrating equipment for reducing wasted waiting time. *CIRP Annals* 66, 1 (2017), 449–452.

[42] Kok-Lim Alvin Yau, Junaid Qadir, Hooi Ling Khoo, Mee Hong Ling, and Peter Komisarczuk. 2017. A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 1–38.

[43] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems* 35 (2022), 24611–24624.

[44] Zhengxu Yu, Shuxian Liang, Long Wei, Zhongming Jin, Jianqiang Huang, Deng Cai, Xiaofei He, and Xian-Sheng Hua. 2021. Macar: Urban traffic light control via active multi-agent communication and action rectification. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 2491–2497.

[45] Xinshi Zang, Huaxiu Yao, Guanjie Zheng, Nan Xu, Kai Xu, and Zhenhui Li. 2020. Metalight: Value-based meta-reinforcement learning for traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1153–1160.

[46] Zijian Zhang, Xiangyu Zhao, Hao Miao, Chunxu Zhang, Hongwei Zhao, and Junbo Zhang. 2023. AutoSTL: Automated Spatio-Temporal Multi-Task Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[47] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. 2019. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1963–1972.

A NOTATIONS

We summarize all notations in this paper and list them in Table 4.

Table 4: Notations in this paper.

Notation	Description
$\mathcal{G}_t$	Dynamic graph at time t
$\boldsymbol{V}_t$	Local node set at time t
v_i	Node i
$\boldsymbol{E}$	Edge set at time t
e_{ij}	An direct edge from node i to node j
N	Total number of signal-controlled intersections
n	Number of nodes in a local graph
m	Number of traffic features
t	Time step in real-world
$\boldsymbol{A}_t$	$n \times n$ adjacency matrix at time t
$\boldsymbol{D}_t$	$n \times n$ degree matrix at time t
$\boldsymbol{X}_t$	$n \times m$ feature matrix at time t
$\mathcal{A}^u$	Action set for agent u
a_t	Action at time t
$\mathcal{S}^u$	State set for agent u
$\boldsymbol{S}_t$	State embedding with dimension k at time t
$\mathcal{R}^u$	Reward set for agent u
r_t	Reward value at time t
$r(\cdot)$	Reward function
π_θ	Policy under with parameter θ
$\boldsymbol{A}_t^{(ij)}$	The weight of the edge e_{ij}
t_{cd}	Count down time
t_{r}	Rest time of green phase
$\mathbf{B}_P^u$	Historical traffic sample set for agent u
$\mathbf{B}_C^u$	Replay buffer for agent u
$\boldsymbol{K}\&\boldsymbol{Q}\&\boldsymbol{V}$	Embedding matrices for attention mechanism
$\mathbf{W}_P$	OTPM's total training parameters
$\theta\&\omega$	OTCM's total training parameters

B MODEL SPECIFICATION

B.1 Derivation of Equation (3)

First, we take Taylor expansion for $e^{q_i^T k_j}$: $e^{q_i^T k_j} = 1 + q_i^T k_j + \frac{(q_i^T k_j)^2}{2!} + \cdots$ and we use the first three terms of it to approximation Equation (2):

$$\text{Att}(q,k,v) \approx \frac{1 + q^{\text{T}}k + \frac{1}{2}\left(q^{\text{T}}k\right)^2}{\sum 1 + q^{\text{T}}k + \frac{1}{2}\left(q^{\text{T}}k\right)^2} v, \tag{14}$$

For symbolic simplicity, we define $x := q^{\text{T}}k$, and we have $\text{Att}(q,k,v) \approx \frac{1+x+1/2x^2}{\sum 1+x+1/2x^2} v$.

Then for a set of fixed x, we can regard Equation (14) as a process of averaging v with $1 + x + 1/2x^2$ as weight and the denominator a constant normalization coefficient c. Defining a scaling function $f(x) = 1 - cx^{-1}$, $y > 0$ and letting $x' := q'^{\text{T}}k'$, we have

$$\text{Att}(q',k',v) \approx \left(1 - \frac{1}{1 + x' + x'^2}\right) v, \tag{15}$$

After that, we replace $1/(1+y), y := x' + x'^2$ by the first two terms of Taylor expansion with respect to y. The left-hand-side of Equation (15) now is to be $\left(x' + x'^2\right) v$. Finally, we write it in a matrix form:

$$\text{Att}(\boldsymbol{Q}', \boldsymbol{K}', \boldsymbol{V}) \approx \boldsymbol{Q}'\boldsymbol{K}'^{\text{T}}\boldsymbol{V} - (\boldsymbol{Q}' \odot \boldsymbol{Q}')(\boldsymbol{K}'^{\text{T}} \odot \boldsymbol{K}'^{\text{T}})\boldsymbol{V}, \tag{16}$$

which is the same as Equation (3).

B.2 Detailed Settings for Numerical Experiments of Attention Module

The node features are generated by Gauss distribution (So the metric of accuracy is only used to compare whether there is a difference between two models). We use an Encoder-Decoder framework to test the newly proposed attention mechanism, as shown in Equation (17).

$$\begin{aligned} \textit{Embedding} &= \text{Encoder}(\text{Att}(Q,K,V)), \\ \textit{Output} &= \text{Decoder}(\text{Att}(\textit{Embedding},K,V)), \end{aligned} \tag{17}$$

where $\boldsymbol{Q}' = \text{Linear}_{W_Q'}(\boldsymbol{X})$, $\boldsymbol{K}' = \text{Linear}_{W_K'}(\boldsymbol{X}), \boldsymbol{V} = \text{Linear}_{W_V}(\boldsymbol{X})$.

B.3 Definition of Reward Function in Section 3.4

In this paper, we define the reward as a piecewise judgment function $r^u := r^u(S_t, S_{t+1})$ for an agent u at the corresponding intersection with z^u lanes. It is the sum of two parts: the mean value of the lane-level rewards of the current green light phase, and the mean value of the lane-level rewards of corresponding to the next green light phase.

$$r = \bar{r}^{\text{g}} + \bar{r}^{\text{ng}} = \frac{1}{x}\sum_{i=1}^{x} r_i^{\text{g}} + \frac{1}{y}\sum_{i=1}^{y} r_i^{\text{ng}}, \tag{18}$$

where x is the number of lanes in the current green light phase, y is the number of lanes in the next green light phase, r_i^{g} and r_i^{ng} are two piecewise judgment functions defined in Equation (19) and (20).

$$r_i^{\text{g}} = \gamma + \begin{cases} -40 & \text{if } \boldsymbol{O}_{t+1}^{(i)} \le L, \boldsymbol{O}_t^{(i)} \le L, \\ -20 & \text{if } \boldsymbol{O}_{t+1}^{(i)} > L, \boldsymbol{O}_t^{(i)} \le L, \\ 10+\beta & \text{if } \boldsymbol{O}_{t+1}^{(i)} \le L, L < \boldsymbol{O}_t^{(i)} \le H, \\ 25+\kappa & \text{if } \boldsymbol{O}_{t+1}^{(i)} \le L, \boldsymbol{O}_t^{(i)} > H, \\ 40 & \text{if } \boldsymbol{O}_{t+1}^{(i)} > H, \boldsymbol{O}_t^{(i)} > H, \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

and

$$r_i^{ng} = \begin{cases} 40 & \text{if } \boldsymbol{O}_t^{(i)} \le L, \boldsymbol{O}_{t+1}^{(i)} \le L, \\ 20 & \text{if } \boldsymbol{O}_t^{(i)} > L, \boldsymbol{O}_{t+1}^{(i)} \le L, \\ -10-\beta & \text{if } \boldsymbol{O}_t^{(i)} \le L, L < \boldsymbol{O}_{t+1}^{(i)} \le H, \\ -25-\kappa & \text{if } \boldsymbol{O}_t^{(i)} \le L, \boldsymbol{O}_{t+1}^{(i)} > H, \\ -40 & \text{if } \boldsymbol{O}_t^{(i)} > H, \boldsymbol{O}_{t+1}^{(i)} > H, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

In both Equation (19) and (20), L and H are two threshold coefficients, $L = 0.3$ and $H = 0.7$, $\boldsymbol{O}_t^{(i)}$ is the time occupancy (Toc) [8] of lane i at time t; β and κ are two congestion penalty factors: if $\boldsymbol{O}_t^{(i)}$ is the maximum value in Equation (19) (or the minimum value in

Equation (20)) among all $O_t^{(j)}$, $j = 1, \cdots, z^u$, then $\beta = 10$ and $\kappa = 5$. Otherwise, $\beta = \kappa = 0$. Besides, γ is a deadlock penalty factor:

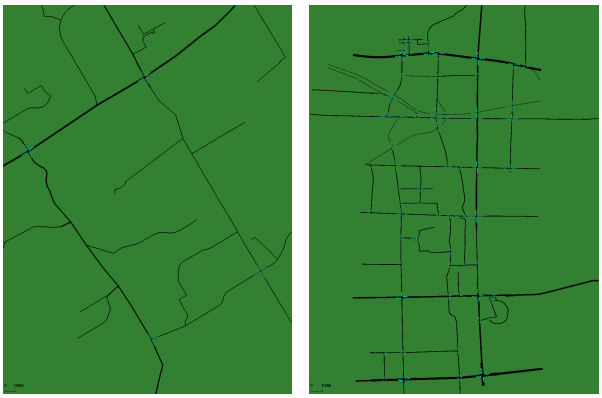
$$\gamma = \begin{cases} -5 & \text{if } O_{t+1}^{(i)} > H, O_{t+1}^{(j)} > H, \\ 0 & \text{otherwise,} \end{cases} \tag{21}$$

where j is the downstream lane of i.

C SIMULATION DETAIL

C.1 Dataset Detail

This paper utilizes two road network datasets, which are visualized in Figure 12. As for the simulated (Sim) road network, we constructed a backbone road network with a well-shaped structure and added a small number of branches on the basis of it.



(a) Sim road network (b) XS road network

Figure 12: Road networks for experiments.

C.2 Baselines Detail

The details of baseline methods used in this paper are described as follows:

- **Fixed** [20] This method uses fixed cycle time and phase time at each intersection for signal control. In this experiment, we configure a distinct timing plan at each intersection for the simulated dataset. We use a real-world traffic timing schedule for the Xiaoshan dataset. It has a 3-6 sub-fixed timing plan for one intersection during a day that is adjusted artificially by signal timing staff.
- **VA (Vehicle Actuated)** [39] This method collects real-time data of queuing vehicles at intersections and mainly uses the longest queue first algorithm for signal control.
- **DynSTGAT** [38] This method adequately exploits the joint relations of spatiotemporal information by using a multi-head graph attention mechanism and efficiently utilize the historical state information of the intersection by designing a sequence model with the temporal convolutional network.
- **FRAP** [47] This method adopts an Ape-X DQN-based distributed framework for traffic signal control, which can converge much faster than existing RL methods during the learning process.
- **Colight** [34] This method first uses graph attentional networks in the setting of Q-learning for large-scale traffic signal control.
- **UniLight** [15] This method first sets up a universal communication form between intersections, which embeds massive observations collected at one agent into crucial predictions of their impact on its neighbors, and then uses Q-Learning to make full use of these communications.
- **HiLight** [40] This method combines two reinforcement learning methods, i.e., Deep Q-Learning (DQN) & clip-Proximal Policy Optimization (clip-PPO), to control traffic lights. DQN is used to deal with different control targets such as travel time and then extract the sub-policies. PPO is used for selecting a specific sub-policy to control each traffic light directly.
- **Macar** [44] This method uses an Actor-Critic algorithm combined with a message propagation graph neural network (MPGNN) based agent communication method for traffic signal control, which helps to rectify action against bias.

C.3 Simulation Environment

Our experimental platform is based on the open-source software, SUMO [2], for secondary development, and is deployed in a Linux server with two Intel(R) Xeon(R) Gold 6248R CPUs, eight NVIDIA TESLA V100 32G graphics cards and 800G memory. In this experiment, the signal control models in all signal machines in the road network are trained and calculated in parallel, and we limit each signal control model only to use an independent CPU thread to simulate the limited computing resources of the roadside units (RSUs).

C.4 Testing Process

Algorithm 2 PRLight Testing Process

Input: Trained parameter $\mathbf{W}_P^u$ for OTPM, trained parameter θ^u for OTCM, real-time observation X_t^u and local-view graph $\mathcal{G}^u$.
Output: Real-time action a_t^u for traffic signal control.
Local Agent u:
1: Observe the real-time traffic feature: X_t^u
2: Calculate dynamic weight $\mathbf{A}_t^u$ and graph embedding H_t^u
3: Predict $X'^u_{t+t_{\text{cd}}}$ based on H_t^u
4: Compute the future state embedding: $S'^u_{t+t_{\text{cd}}}$
5: Sample an control action a_t^u by policy: $\pi_{\theta^u}(a_t^u | S'^u_{t+t_{\text{cd}}})$
6: **return** a_t^u to the traffic signal machine for signal control

As an extension to section 3.5, this section continues to introduce the testing process of the framework. During the testing period, we have the trained parameters $\mathbf{W}_P$, θ^u, ω^u. Then for an agent u, we apply OTPM and OTCM to calculate an action a_t^u by giving the real-time observation X_t^u and A_t^u, as shown in Algorithm 2.

[2] https://www.eclipse.org/sumo/