DuaLight: Enhancing Traffic Signal Control by Leveraging Scenario-Specific and Scenario-Shared Knowledge

Jiaming Lu[†‡]
SenseTime Research
ISTBI of Fudan University
Shanghai, China
lujia_ming@126.com

Jingqing Ruan[†‡]
SenseTime Research
Beijing, China
ruanjingqing@sensetime.com

Haoyuan Jiang[§]
SenseTime Research
Baidu Inc.
Shenzhen, China
jianghaoyuan@zju.edu.cn

Ziyue Li[*]
University of Cologne
EWI gGmbH
Cologne, Germany
zlibn@wiso.uni-koeln.de

Hangyu Mao
SenseTime Research
Beijing, China
maohangyu@senstime.com

Rui Zhao
SenseTime Research
Qing Yuan Research Institute of
Shanghai Jiao Tong University
Beijing, China
zhaorui@sensetime.com

ABSTRACT

Reinforcement learning has been revolutionizing the traditional traffic signal control task, showing promising power to relieve congestion and improve efficiency. However, the existing methods lack effective learning mechanisms capable of absorbing dynamic information inherent to a specific scenario and universally applicable dynamic information across various scenarios. Moreover, within each specific scenario, they fail to fully capture the essential empirical experiences about how to coordinate between neighboring and target intersections, leading to sub-optimal system-wide outcomes.

Viewing these issues, we propose DuaLight, which aims to leverage both the experiential information within a single scenario and the generalizable information across various scenarios for enhanced decision-making. Specifically, DuaLight introduces a scenario-specific experiential weight module with two learnable parts: Intersection-wise and Feature-wise, guiding how to adaptively utilize neighbors and input features for each scenario, thus providing a more fine-grained understanding of different intersections. Furthermore, we implement a scenario-shared Co-Train module to facilitate the learning of generalizable dynamics information across different scenarios. Empirical results on both real-world and synthetic scenarios show DuaLight achieves competitive performance across various metrics, offering a promising solution to alleviate traffic congestion, with 3-7% improvements. The code is available under https://github.com/lujiaming-12138/DuaLight.

† These authors contribute equally to this work.
‡ These authors work as research interns at SenseTime Research.
§ Work done at SenseTime Research, now in Baidu Inc.
∗ The corresponding author.

1 INTRODUCTION

Traffic congestion has emerged as a pressing issue in metropolises, leading to protracted travel and waiting time, heightened energy consumption, and diminished commuting comfort [1–4]. Consequently, traffic signal control (TSC) has increasingly become a focal point of research, presenting an efficacious approach to alleviating such urban gridlock [5].

Recently, the paradigm of TSC has predominantly shifted towards deep reinforcement learning (RL) [6]. Such learning-based approaches [7, 8] can "learn" to give optimal actions directly based on the observation of intersections, which has proved its superiority over the conventional traffic-engineering-based methods such as SCATS and SCOOT [9–12], which are static models based on assumptions that could be unrealistic in front of the traffic dynamics. Currently, there are two state-of-the-art solutions emerging in RL-based TSC: (1) cooperation among multi-agents and (2) learning via multi-scenarios.

The first focuses on the cooperation of multiple intersections in one single *scenario* (a simulation environment containing a set of intersections). With each intersection as an agent, multi-agent RL (MARL) [13, 14] have been developed. Most of the MARL-based TSC try to advocate cooperation by aggregating the information of the agents: they integrate the state of the target intersection with its neighboring intersections' states, either spatially [15, 16] or spatiotemporally [17, 18], based on GNN [16] or GNN+LSTM/TCN to additionally capture long-range dependency [18], respectively. Despite their potential, these single-scenario-based approaches ignore that **how to coordinate with neighbors differs from**
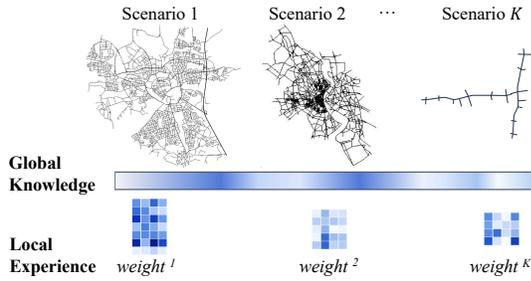
Figure 1: (1) Global knowledge from co-training, and (2) scenario-specific experiential knowledge formed as weights.

scenarios: e.g., collaborating with 1-hop neighbors works the best in simple scenarios, while 2-hop neighbor collaboration is needed for more complex scenarios. Thus, training the model in only one scenario may lead to a local optimum due to overfitting [19].

The second type focuses on learning across multiple scenarios, such that the model could be more general for various regions or cities. To achieve this, different techniques such as meta RL [20, 21], attention mechanism [22] and standardization of intersections [19] have been proposed. For instance, MetaLight [21] proposed a meta gradient learner through different datasets, and GESA [19] proposed a plug-and-play mapping module to enable multi-scenario co-training. These methods offer a potential solution to the overfitting problem mentioned before. However, they are all single-agent based, meaning one agent controls all the intersections, which may be an easy start for multi-scenario learning. MetaGAT [23] extended MetaLight to the multi-agent version by simply adding GAT in multi-scene training. However, these methods overlook that **how to utilize the unique knowledge of each scenario to facilitate the cooperation**: can we design an explicit mechanism for modeling experiential information within a single scenario?

To tackle these challenges, we propose **DuaLight** with two modules: **scenario-shared co-train** and **scenario-specific experiential weights**. As shown in Fig 1: (1) Some knowledge about the traffic underlining mechanisms is universal and commonly shared across various scenarios, for example, the flow is periodic, there are the potential morning/evening peak, and the merging and diversion of traffic flow along the network affect the traffic. Co-Train module enables multi-scenario joint learning of such global knowledge. To encourage stability, only a **subset** of the model's parameters is trained concurrently, yet the essential coordination parameters are learned within each scenario. (2) Some knowledge is scenario-specific: for example, each scenario has its unique distribution of the traffic flow, some tend to have more morning peaks and others more evening peaks, etc. This will affect different cooperation patterns from traffic lights. The experiential weight module defines the coordination parameters as the intersection-wise and feature-wise weights, guiding how to aggregate neighboring intersections' information and different observation features, respectively. The two weights are trained after observing a whole episode, capturing the long-term experiences.

The combination of these two modules encourages the model to learn and balance the shared dynamic information across multiple scenarios and the dynamic experiential information within a particular scenario, thus enabling the model to learn an effective representation to assist decision-making. This simple and effective dual design also supports our extension of using the neighbors **even from other scenarios**, which is a novel and promising discovery, as it can learn from similar intersections in other scenarios to further improve the ability of signal control.

In summary, this paper has three main contributions:

- To the best of our knowledge, we are the first that considers both scenario-common and scenario-specific information by co-train module and experiential weight module, respectively. This design also enables us to discover the potential of aggregating neighbors across scenarios. Overall, we coordinate multi-agents better across multi-scenario.
- Specifically, we design the scenario-specific experiential weights that encourage modeling the influence of neighbors and input features, adaptive to different scenarios.
- We conduct experiments in both real-world and synthetic scenarios: DuaLight has the fastest and the most stable training and achieved the best results with 3-7% improvements.

2 RELATED WORK

Learning to cooperate. In the realm of RL-based TSC, [24, 25] directly train a centralized agent by using the observations of all intersections in a scenario as input to the model and providing a decision for each intersection. However, the complexity of these methods increases as the number of intersections increases, and it is hard to explore and optimize due to the curse of dimension in joint action space. To ease this complexity, many MARL models take each intersection as an agent [26], with surrounding intersections considered for better decisions. For example, CoLight [16] and MetaGAT [23] employed GAT to assign varied weights to neighboring intersections. Yet, this approach primarily views neighboring intersections' information from a spatial standpoint in an instant short-sighted manner, without considering the influence of historical experiences on decision-making. To consider temporal information, STMARL [17] and DynSTGAT [18] proposed to use LSTM or TCN to capture the historical state information, e.g., traffic flow, and employs GNN or GAT to extract the spatial dependencies. However, these methods only considered the temporal dependency of state $(s_1, s_2, s_3, ...)$, such as traffic flow; our experiential module instead allows for explicit capture of dynamic sequences $(s_1, a_1, r_1, s_2, a_2, r_2, ...)$ through gradient propagation, offering a more comprehensive and nuanced understanding of scenario-specific decision dynamics. Moreover, as mentioned before, these methods neglect that different scenarios will have different collaboration patterns due to various network structures and traffic dynamics. Our Experiential Weight uniquely adapts to varying scenario impacts through learnable weights. Unlike methods limited to single-scenario, DuaLight dynamically captures both unique and shared traffic dynamics across scenarios, enhancing adaptability and insight through backpropagation.

Learning across multiple scenarios. Simultaneously, some methods examine training in multi-scenario for optimized performance. The single-agent version is mostly dominating. MetaLight
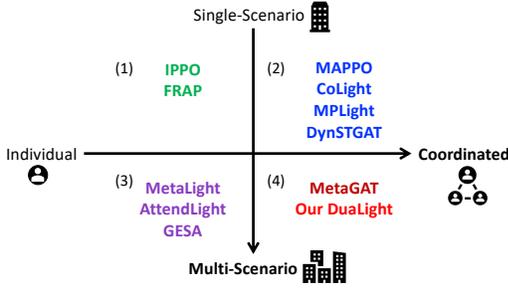
Figure 2: Two state-of-the-art methods: (1) whether it's multi-agent and (2) whether it's multi-scenario.

[21], based on Meta RL [27], permitted joint scenario training and direct transfer to new scenarios. Similarly, GeneraLight [20] enhanced generalization by merging various traffic flows with generative adversarial networks and the MetaRL. AttendLight [22] introduced a training framework suitable for intersections of varying traffic flows and configurations by utilizing two attention models. The most recent GESA [19] presented a universal intersection normalization scheme and leveraged the A3C algorithm for joint training across multiple scenarios. However, a single agent controlling all is not optimal. Yet simply putting a multi-agent model in a multi-scenario learning setting could experience devastatingly unstable training, which has already taken shape in the single-agent MetaLight. MetaGAT [23], as the most related work, tried to coordinate multi-agents across multi-scenario: It stabilized the training by separate task encoding (based on GAT+TCN) and controlling. Similarly, we stabilize the learning by only training part of the parameters concurrently, and the scenario-specific experiential weights are learned separately. These weights not only explicitly preserve the long-term MDP experiences of each scenario, guiding a scenario-adaptive GAT and input-feature aggregation, but also enable a simple extension to use neighbors even across scenarios.

3 PROBLEM DEFINITON

Before introducing the model, we shortly recap some key concepts integral to TSC. We recommend referring to [19] for more details.

DEFINITION 3.1 (INTERSECTION). *An **intersection** I_i is where multiple roads connect and are controlled by a traffic light. A standard intersection, shown in Fig. 3(a), consists of four **entrance arms** (N, S, E, W), each containing three possible entrance lanes: left-turn, through, and right-turn (also known as **traffic movements**). Each entrance arm has an exit arm as an outlet for vehicles. The majority of intersections are either with 3-arm or 4-arm structures.*

DEFINITION 3.2 (PHASE). *A traffic phase is a combination of traffic movements in which there is no conflict between them. Fig.3(b) depicts eight phases in a standard intersection. In the setting of RL in TSC, the action space $\mathcal{A}$ of the agent refers to select the phase.*

We reconceptualize the TSC problem as a Partially Observable Markov Decision Process (POMDP) since each agent only observes part of the whole global state of the city.

PROBLEM STATEMENT 3.1. ***TSC as POMDP**: TSC is a sequential decision-making problem. Assuming there are N intersections in a*
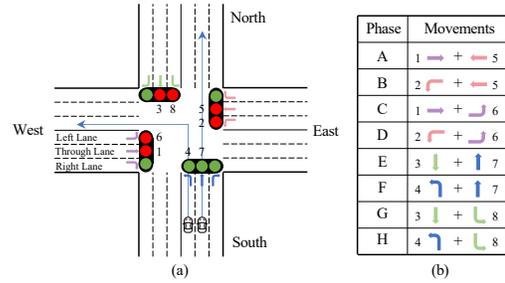
Figure 3: (a) A 4-arm intersection with eight traffic movements (1-8), presently controlled by phase F. (b) Eight phases (A-H), each having two non-conflicting traffic movements.

scenario, each intersection is controlled by an independent agent. The agent's goal is to learn a signal control policy to optimize travel time, which can be formulated as a POMDP $\langle \mathcal{S}, O, \mathcal{A}, \mathcal{P}, r, \gamma, \pi \rangle$.

System state space $\mathcal{S}$ and **Partial observation space O**: At the time t, each agent can observe a local observation $o_i^t \in O$ from the global system state $s^t \in \mathcal{S}$, including the current phase and the number of stopped vehicles on the road.

Action space $\mathcal{A}$: $\mathcal{A}$ of each agent is to select one of the eight traffic phases (A-F) shown in Fig. 3: Based on o_i^t, each agent selects an action $a_i^t \in \mathcal{A}$ as the traffic signal control logic for the next time interval Δt.

Transition probability $\mathcal{P}$: A function for the system to enter the next state s^{t+1}, which is defined as $\mathcal{P}\left(s^{t+1} \mid s^t, a^t\right)$. This is an unknown function that encapsulates the dynamic information of traffic system operations.

Reward function r: After executing a phase, a reward can be obtained from the system based on the reward function. The immediate reward of agent i at time t: $r_i^t = -w \sum_l q_{i,l}^t$, in which, $q_{i,l}^t$ represents the number of stopped vehicles on the approaching lane l at time t, and w represents the punishment coefficient, we set it as 0.25.

Policy π: a agent's controlling policy. At each time t, each agent follows policy $\pi(a^t \mid o^t)$ to make an action a^t based on the current observation o^t, with the objective of minimizing all rewards $G_i^t = \sum_{t=\tau}^{T} \gamma^{t-\tau} r_i^t$, where γ is the **discount factor** ($\gamma = 0.95$).

4 METHODOLOGY

In the subsequent section, we delineate our proposed DuaLight, an end-to-end MARL architecture, as depicted in Fig. 4. We will first introduce the feature extraction module, followed by the Scenario-Specific Experiential Weight Module and its coordination with GAT to aggregate observation information within neighbor intersections and self-features. Then, we will introduce the Scenario-Shared Co-train Module and the objective function for training.

4.1 Feature Extractor

In this stage, we first obtain feature representations from the simulator's raw observations of all lanes at the intersection, including the number of cars and the current stage. The multi-layer perceptron (MLP) is applied as the feature extractor $fe(\cdot)$ in the following.

$$\psi(o^i) = fe(o^i) = \sigma(o^i W_f + b_f), \quad (1)$$
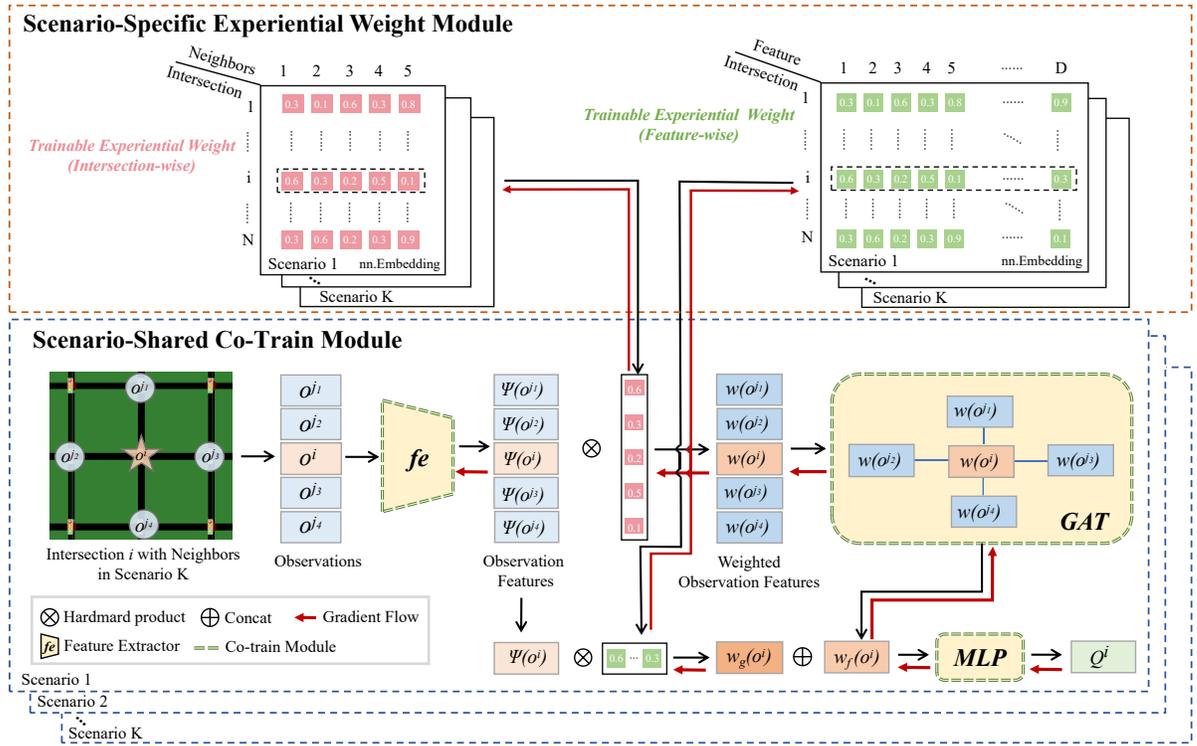
Figure 4: The illustration of our proposed DuaLight. The upper part of the figure represents the trainable experimental weights. The lower part gives the detailed interference process of all the modules.

where $o^i \in \mathbb{R}^F$ is the raw observation of the intersection i with the dimension of F, $W_f \in \mathbb{R}^{F \times D}$, $b_f \in \mathbb{R}^D$ are the weight matrix and the bias of the MLP, and $\sigma(\cdot)$ denotes the *ReLU* activation function.

Thus, we obtain a D-dimensional representation as the base feature for each intersection. Next, we present the experiential weight to process these features further.

4.2 Scenario-Specific Experiential Weight

Merely projecting raw observations is often inadequate for traffic light control, as it requires a long-term and experiential understanding of both intersection-wise and self-feature-wise perceptions. Specifically, the ability to perceive intersection-wise information is crucial for the decision-making process of an intersection, as it enables effective coordination between multiple intersections, leading to improved traffic flow and reduced congestion throughout the road network. Additionally, self-feature-wise perception is also essential for accurate decision-making that can alleviate congestion at the current intersection. Thus, we propose the experiential weight mechanism, which enables keeping the neighbor-intersection-wise and the self-feature-wise memory throughout the training process.

*4.2.1 **Two Trainable Experiential Weights**.* Here, we define the trainable experiential weight matrices as intersection-wise $\{Emb_{int}^k\}_{k=1}^K$, and feature-wise $\{Emb_{fea}^k\}_{k=1}^K$, where for a given scenario k, $Emb_{int}^k \in \mathbb{R}^{N_k \times (1+N_{nei})}$, $Emb_{fea}^k \in \mathbb{R}^{N_k \times D}$. K, N_k, N_{nei} denote the number of scenarios, the number of intersections in the

scenario k, and the number of neighbors of an intersection, respectively. Moreover, to ensure precise attention to an agent's own features, it is crucial to assign a separate feature weight to each scenario k. These two weight matrices are updated at each iteration during the training phase and are fixed during inference.

In the implementation, the trainable experiential weight matrix corresponds to a special MLP layer without bias and non-linear activation function, implemented as a PyTorch module using *nn.Embedding*. During the training, these embeddings can be updated using the gradient $\nabla_\phi \mathcal{L}$ of optimization objective $\mathcal{L}$ (details in Eq. (13)) in an end-to-end manner, which enables these modules to contain intersection-wise and feature-wise representative historical information from the experiential replay buffer.

$$Emb_{\{int,fea\}}^k \leftarrow Emb_{\{int,fea\}}^k - \alpha \nabla \mathcal{L}_\phi \qquad (2)$$

Next, we introduce the acquisition process of these weights, elaborated as follows.

1) The acquisition of intersection-wise experiential weight. To compute the intersection-wise experiential weight for each agent i, we begin by setting $N_{nei} = 4$ and $\mathcal{N}_i = \{j_1, j_2, j_3, j_4\}$. Here, we assume that there are four neighbors that can be found through the nearest distance metric, and this assumption holds for the rest of the discussion. In Sec. 6, we relax it by taking more neighbors, even from other scenarios. To obtain the intersection-wise experiential weight, we use the operator $lookup(\mathbf{X}, i)$ to return the i-th row of

X:

$$weight_{int}^{k,i} = lookup(Emb_{int}^k, i), \qquad (3)$$

where $weight_{int}^{k,i} \in \mathbb{R}^{1+N_{nei}}$ and Emb_{int}^k is trained in an end-to-end manner according to the RL target. Thus, as the training process progresses, the weight is endowed with high-level semantic information that represents the degree of attention between agents and their neighbors in the long run. This simple yet effective design (storable embedding + readout) allows a good extendability when even considering the neighbors from other scenarios (in Sec. 6).

2) The acquisition of feature-wise experiential weight. For each scenario k and each intersection i, we can obtain the feature-wise experiential weight as follows.

$$weight_{fea}^{k,i} = lookup(Emb_{fea}^k, i) \qquad (4)$$

where $weight_{fea}^{k,i} \in \mathbb{R}^D$. Similarly, as the training process continues, the model learns the optimal weight for each intersection feature, allowing it to assign appropriate attention to each feature during decision-making. The feature weight is critical to ensure the model adapting to varying scenarios and making reliable predictions, as each scenario may require a different emphasis on certain features.

In summary, the model's two weights have distinct roles. The intersection-wise weight linearly transforms input features before the GAT layer, capturing complex feature relationships and improving the model's learning and generalization. The feature-wise weight ensures reliable predictions by adapting to different scenarios, each requiring emphasis on specific features.

Next, we will introduce how to integrate the experiential weights into the decision-making process.

4.2.2 Scenario-Specific Knowledge Injection. As shown in the lower part of Fig. 4, there involves scenario-specific knowledge injection in the decision-making process.

1) The knowledge from the neighboring intersection. At the intersection-wise level, we take the experiential (global) and instant (local) impacts into consideration. The global experiential impact can be perceived through the intersection-wise weight, and the weighted observation feature of agent i and its neighbors $N_i \in \mathbb{R}^{N_{nei}}$ can be represented as follows.

$$w(o^i) = weight_{int}^i \otimes \psi(o^i) \qquad (5)$$

$$\{w(o^j)\}_{j \in N_i} = weight_{int}^j \otimes \psi(o^j), \qquad (6)$$

where $\otimes$ denotes the Hardmard product.

Then we extract the local instant impact via GAT [16, 28] due to its powerful representation capacity. Through the attention mechanism in GAT, the important coefficients a^{ij} are computed:

$$a^{ij} = \frac{\exp(w(o^i)\widehat{W}(w(o^j)\widehat{W})^{\mathsf{T}}/\sqrt{D'})}{\sum\limits_{l \in N_i} \exp(w(o^i)\widehat{W}(w(o^l)\widehat{W})^{\mathsf{T}}/\sqrt{D'})}, \sum_{j \in N_i} a^{ij} = 1, \qquad (7)$$

where l iterates among the set containing the agent i and its neighbors N_i, $\widehat{W} \in \mathbb{R}^{D \times D'}$ is a learnable weight matrix for the attention mechanism, D' denotes the dimension of the latent vector. Moreover, multi-head attention (MHA) is used to stabilize the training process. We apply the average pooling to the hidden vectors from each head and pass through a transformation to produce the final

---

Algorithm 1: The Pseudo-code of DuaLight

ensure : the co-training networks $C = \{fe, GAT, MLP\}$, and the experiential weights $Emb = \{Emb_{int}, Emb_{fea}\}$

initialize: $L, T, K, N_k, \mathcal{B}$; // The number of training episodes, the number of timesteps in an episode, the number of scenarios, the number of intersections in scenario k, and the experience replay buffer;

initialize: the parameter θ for the co-training networks C, and ϕ for the experiential weights Emb;

1 **for** *episode* $l = 1$ **to** L **do**
2   **for** *timestep* $t = 1$ **to** T **do**
3   **for** *scenario* $k = 1$ **to** K **do**
4   **for** *intersection* $i = 1$ **to** N_k **do**
5   Observe $o_t^{k,i}$;
6   Get the experiential weights by Eq. (3), and Eq. (4);
7   Obtain action $a_t^{k,i}$ by arg max Q^i of Eq. (10);
8   Receive the reward $r_t^{k,i}$ and the next observation $o_{t+1}^{k,i}$;
9   Store the transition $(o_t^{k,i}, a_t^{k,i}, r_t^{k,i}, o_{t+1}^{k,i})$ in $\mathcal{B}$;
10   Sample a minibatch $\mathcal{M}_1$ from $\mathcal{B}$;
11   Update the experiential weights Emb using Eq. (14a) with $\mathcal{M}_1$;
12   Sample a minibatch $\mathcal{M}_2$ from $\mathcal{B}$;
13   Update the co-train networks C using Eq. (14b) with $\mathcal{M}_2$;

---

output. Thus, the final latent feature $w_g(o^i)$ aggregates i's neighbors' information into i by adopting GAT:.

$$w_g(o^i) = \sigma\left(\frac{1}{M} \sum_{m=1}^M \sum_{j \in N_i} a_m^{ij} W^m w(o^j)\right), \qquad (8)$$

where $\sigma(\cdot)$ is the activation function, M is the number of attention heads, a_m^{ij} is the attention score of the m-th attention head in Eq. (7), W^m is the learnable matrix with respect to the head m.

2) The knowledge from the self input feature. At the feature-wise level, the feature $\phi(o^i)$ of the agent i is weighted by the feature-wise weight from Sec. 4.2.1.2), calculated as follows.

$$w_f(o^i) = weight_{fea}^{k,i} \otimes \phi(o^i) \qquad (9)$$

At the end, the multi-source message is contacted and passed through an MLP layer to predict the final state-action value.

$$\begin{aligned} Q^i &= MLP(w_g(o^i) \oplus w_f(o^i)) \\ &= (w_g(o^i) \oplus w_f(o^i))W^o + b^o \end{aligned} \qquad (10)$$

where $p = 8$ is the dimension of the action space, i.e., the number of phases. $\oplus$ denotes the concat operator, $W^o \in \mathbb{R}^{2D \times p}$ and $b^o \in \mathbb{R}^p$ are the weight matrix and the bias vector of the output MLP layer.

4.3 Scenario-Shared Co-Train Module

To encourage the model to learn the common patterns that are independent of scenarios, we aim to co-train various scenarios together. The difficulties are: most RL-based TSC models assume homogeneity across intersections, i.e., equalities in observation space, action space, reward function, and policy π. However, the standard 4-arm intersection structure does not ubiquitously apply in the real world. There are 4-arm intersections with irregular angles and also 3-arm or 5-arm intersections. Different cities definitely have different intersections with different structures (with different numbers of entrance arms, different combinations of lanes). This observation

necessitates the standardization of intersections, translating non-standard forms into a uniform 4-arm structure. To achieve this, we deploy a mapping approach in [19]: GESA uses the relative orientation of entrance arms (e.g., the relative angels) to decide the conflicting movements, instead of the absolute orientation (N, S, E, W). The "missing" entrance arm's corresponding state is masked, and its action is zero-padded. More detail is given in [19].

Contrasting the approaches by GESA [19] and other MetaRL-based methods [21, 23, 29], where all parameters of the model undergo joint training across multiple scenarios, our Co-Train module only allows the parameters within the **fe, GAT**, and **MLP** in Fig. 4 to participate in such joint training. Conversely, the Experiential Weight module employs data exclusively from a single scenario for their training. This mechanism enables the Experiential Weight module to concentrate more effectively on capturing information within a specific scenario, whereas the **fe, GAT**, and **MLP** modules focus on grasping the general information across various scenarios.

During co-training, we use multi-processing, with each process using SUMO [30] for interaction across different scenarios and subsequently aggregating all data from each process into a unified buffer. When we sample data from the buffer for model updating, data from different scenarios are sampled with equal weight. During the network update, the parameters of the three modules—**fe, GAT**, and **MLP**—are updated using data from all scenarios.

Our results demonstrate that the incorporation of the Multi-Scenario Co-Train module expedites the convergence of the model. When paired with the Experiential Weight module, the model not only becomes more stable but also delivers improved performance.

4.4 Training Objective

We adopt the value-based reinforcement learning regime to define the loss. The parameter-sharing mechanism is applied across all the agents. For scenario k, the objective is to find the optimal Q-function that maximizes the expected return.

$$Q^k(s_t, a_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma r_t^k | s_t, a_t\right], \tag{11}$$

where Q^k is the action-value function for the scenario k, s_t and a_t are the state and action at time step t, r_t is the immediate reward received after taking action a_t, and γ is the discounted factor.

At the time t, we can compute target Q value as below:

$$Q_{target}^k(s_t, a_t) = r_t^k + \gamma \max_{a'} Q^{k-}(s_{t+1}, a'), \tag{12}$$

where Q^{k-} is the target network.This target network is a copy of the main network that is used to calculate the Q-values during training, but its parameters are not updated during the learning process. Instead, the target network's parameters are periodically updated with the parameters of the main network, which helps to stabilize the learning process and prevent oscillations or divergence.

Next, using Stochastic Gradient Descent (SGD) to approximate the gradient of Q-learning and compute the loss and its gradient, we can write down the following rules.

$$\mathcal{L} = \frac{1}{2}\|Q^k(s_t, a_t) - Q_{target}^k(s_t, a_t)\|^2,$$
$$\nabla_{\theta,\phi}\mathcal{L} = (Q^k(s_t, a_t) - Q_{target}^k(s_t, a_t)) \nabla_{\theta,\phi} Q^k(s_t, a_t), \tag{13}$$

Now we can update the parameters by

$$\theta \leftarrow \theta - \alpha \cdot \nabla_{\theta}\mathcal{L}, \quad \phi \leftarrow \phi - \alpha \cdot \nabla_{\phi}\mathcal{L}, \tag{14}$$

where α is the learning rate, θ denotes the parameter for the co-training networks {**fe, GAT, MLP**}, and ϕ denotes the experiential weights $Emb_{\{int,fea\}}$. The model is summarized in Algorithm 1.

5 EXPERIMENTS

In this section, we outline the configuration of our experiments, the dataset, the comparative methods, and design multi-dimensional experiments to verify the effectiveness of our proposed DuaLight.

5.1 Experiment Settings

Environment Setting: For performance evaluation, we adopt the Simulation of Urban Mobility (SUMO)[1], extensively acknowledged and embraced by both academia and industry, as our experimental simulation platform. Within this simulated framework, each individual simulation proceeds for a duration of 3600 seconds, with the model making its decisions at an interval of Δt = 15 seconds.

Model Setting: We provide the detailed hyper-parameter settings in Table A2 of Appendix B.

5.2 Datasets

Our model is assessed using three synthetic datasets and four datasets derived from real-world scenarios, summarized in Table A1 of Appendix A. Synthetic Datasets include *Grid* 4×4 [31], *Avenue* 4×4 [32], and *Grid* 5×5. Real-world Datasets include *Cologne8* [33] and *Ingolstadt21* [34] from Germany, as well as *Fenglin* and *Nanshan* [19] from China. For more details, please refer to [19, 35].

5.3 Compared Methods

DuaLight is compared with two distinct categories of signal control models: the first category is traditional traffic-engineering-based models and the second is RL-based models.

Traditional Methods:

- **Fixed-timed Control (FTC)** [9]: This method employs expert knowledge to manually assign fixed phase sequences and durations to each traffic signal.
- **MaxPressure** [36, 37]: The pressure at each intersection is estimated by gauging the number of vehicles and queue length. Subsequently, the algorithm invariably selects phases that maximize this pressure in a greedy manner.

Reinforcement Learning-based Methods:

- **IPPO** [35, 38]: In independent PPO agents, each traffic signal is modeled as an independent agent. They utilize the same network architecture, but their parameters are not shared.
- **MPLight** [31]: This algorithm is based on the phase competition FRAP framework [39] and employs pressure as both state and reward for the DQN agents.
- **MetaLight** [21]: It integrates the FRAP framework with Meta RL to facilitate swift adaptation to new scenarios and enhance overall performance. This algorithm bears similarities with our proposed Multi-Scenario Co-train module.

---

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

Methods	Avg. Trip Time (seconds)							Avg. Delay Time (seconds)						
	Grid4×4	Grid5×5	Arterial4×4	Ingolstadt21	Cologne8	Fenglin	Nanshan	Grid4×4	Grid5×5	Arterial4×4	Ingolstadt21	Cologne8	Fenglin	Nanshan
FTC	206.68 (0.54)	550.38 (8.31)	828.38 (8.17)	319.41 (24.48)	124.4 (1.99)	344.76 (6.84)	729.02 (37.03)	94.64 (0.43)	790.18 (7.96)	1234.30 (6.50)	183.70 (26.21)	62.38 (2.95)	283.13 (12.78)	561.69 (37.09)
MaxPressure	175.97 (0.70)	274.15 (15.23)	686.12 (9.57)	375.25 (2.40)	95.96 (1.11)	316.01 (4.86)	720.89 (29.94)	64.01 (0.71)	240.00 (18.43)	952.53 (12.48)	275.36 (14.38)	31.93 (1.07)	372.08 (267.2)	553.94 (32.61)
MPLight	179.51 (0.95)	261.76 (6.60)	541.29 (45.24)	319.28 (10.48)	98.44 (0.62)	329.81 (4.19)	668.81 (7.92)	67.52 (0.97)	**213.78** (14.44)	1083.18 (63.38)	185.04 (10.70)	34.38 (0.63)	399.34 (248.82)	494.05 (7.52)
IPPO	167.62 (2.42)	259.28 (9.55)	431.31 (28.55)	379.22 (34.03)	90.87 (0.40)	368.14 (6.25)	743.69 (38.9)	56.38 (1.46)	243.58 (9.29)	914.58 (36.90)	247.68 (35.33)	26.82 (0.43)	324.57 (12.19)	577.99 (42.22)
rMAPPO	164.96 (1.87)	300.90 (8.31)	565.67 (44.8)	453.61 (29.66)	97.68 (2.03)	412.73 (14.54)	744.47 (30.07)	53.65 (1.00)	346.78 (28.25)	1185.2 (167.48)	372.2 (39.85)	33.37 (1.97)	403.6 (57.29)	580.49 (33.6)
CoLight	163.52 (0.00)	242.37 (0.00)	409.93 (0.00)	337.46 (0.00)	**89.72** (0.00)	324.2 (0.00)	**608.01** (0.00)	51.58 (0.00)	248.32 (0.00)	776.61 (0.00)	226.06 (0.00)	25.56 (0.00)	262.32 (0.00)	**428.95** (0.00)
MetaLight	169.21 (1.26)	265.51 (10.53)	424.39 (24.49)	349.89 (2.65)	97.93 (0.74)	316.57 (4.29)	653.23 (9.15)	57.56 (0.76)	270.06 (31.54)	873.28 (39.01)	227.48 (4.25)	29.01 (0.69)	376.11 (244.85)	478.81 (10.29)
MetaGAT	165.23 (0.00)	266.60 (0.00)	**374.80** (0.87)	290.73 (0.45)	90.74 (0.00)	**290.73** (0.45)	676.42 (0.00)	53.20 (0.00)	234.80 (0.00)	772.36 (0.00)	264.07 (9.85)	26.85 (0.00)	**176.86** (2.37)	503.42 (0.00)
DuaLight	**161.04** (0.00)	**221.83** (0.00)	396.65 (0.00)	317.97 (0.00)	89.74 (0.00)	313.22 (4.88)	609.89 (0.00)	**49.32** (0.00)	237.71 (0.00)	756.99 (69.44)	**182.67** (9.34)	**25.35** (0.00)	260.87 (0.00)	429.49 (0.00)

Table 1: Performance of synthetic and real-world data, including the mean and standard deviation (in parentheses). Best results in boldface, and the second best results underlined.

- **rMAPPO** [14, 40]: It is a widely adopted MARL framework with an actor-critic architecture that leverages proximal policy optimization to boost the stability of training. In this instance, we employ a variant equipped with an RNN module to encode historical information.
- **CoLight** [16]: It utilizes a GAT to aggregate the state information of neighboring intersections.
- **MetaGAT** [23] utilizes GAT-based context to enhance collaborative interactions between intersections. It is also a multi-agent multi-scenario method.

5.4 Evaluation Metrics

Consistent with [35], we utilize **Average Delay, Average Trip Time**, and **Average Waiting Time** as evaluation metrics to measure the efficacy of the various TSC models. Among them, Delay represents the delay caused by signalized intersections (stop or approach delay), Trip Time represents the total time for a vehicle to travel from its starting point to its destination, and Waiting Time represents the time spent by a vehicle waiting at the intersections.

5.5 Main Results

In this section, we introduce the results yielded by DuaLight and other methods based on the various evaluation metrics.

Tables 1 present the outcomes of our proposed DuaLight algorithm in comparison to other traditional control algorithms and RL-based algorithms. DuaLight exhibits optimal performance on the majority of indicators or holds competitive outcomes in relation to the best-performing algorithm.

Among all scenarios, DuaLight demonstrates the most substantial enhancement in the Average Waiting Time metric, with an average improvement of 4.52%, 6.86% on the synthetic dataset, and 2.76% on the real-world scenario. This is owing to our scenario-specific experiential weight design and scenario-shared multi-scenario co-training. Despite DuaLight achieving SOTA results on the Delay metric barring *Grid* 5 × 5 and *Nanshan*, its average performance is not as exceptional due to its less impressive performance on *Grid* 5×5 compared to MPLight. Looking at the TripTime metric, *Fenglin, Ingolstadt21*, and *Grid* 5 × 5 all achieved significant improvements, indicating that DuaLight can handle complex scenarios. Full metrics evaluation is shown in Table A3 of Appendix C.

5.6 Embedding Visualization of DuaLight

To ascertain how our proposed scenario-specific experiential weight module is utilized, and to explore the information learned by it, we visualize the embeddings post message aggregation using t-SNE

Metric	All	Synthesized	Real-world
Avg. Delay	-0.35%	-1.43%	0.45%
Avg. Trip Time	2.03%	4.41%	0.25%
Avg. Wait	4.52%	6.86%	2.76%

Table 2: Improvement of DuaLight across different scenarios

(a1) Ours: Round 1 **(a2) Ours: Round 40** **(a3) Ours: Round 80**

(b1) CoLight: Round 1 **(b2) CoLight: Round 40** **(b3) CoLight: Round 80**
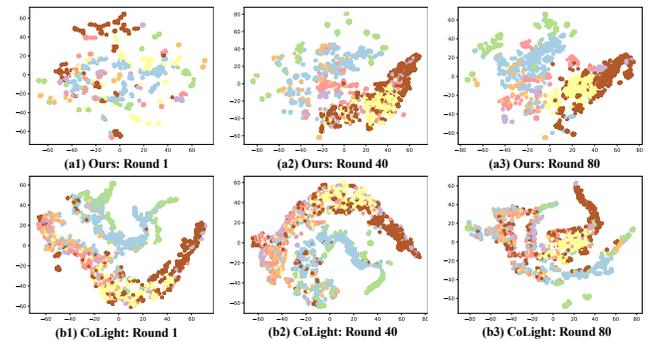
Figure 5: The t-SNE visualization of the hidden embeddings after message aggregation

[41] after different rounds. We independently repeat the evaluation five times, and each time we select time steps 100-110 for visualization. For each agent, we extract its weighted embeddings before the MLP in Fig. 4, and visualize them via t-SNE, shown in Fig. 5.

A point in Fig. 5 represents an agent, and different colors represent agents from disparate scenarios. From Fig. 5(a1) to (a3), we observe that, after a certain number of training iterations, DuaLight's agent embeddings (weighted by experiential module) from the same scenario are moving closer to form one cluster. This suggests that our experiential weight module assists agents in capturing information within a certain scenario. While the embeddings weighted by GAT for CoLight, shown in Fig. 5(b1) to (b3), reveal that even after numerous training iterations, CoLight remains unable to distinguish differences between scenarios.

This is owing to our learnable Intersection-wise and Feature-wise modules participating in each round of model updating. Through continuous iterations of model updating, these modules can aggregate historical and environmental experiential information and extract the dynamic characteristics of neighbors and self-feature information in the corresponding scenarios. More specifically, Intersection-wise can aid agents in comprehending the long-term impact of surrounding neighbors, while Feature-wise can assist agents in understanding the significance of different features. Conversely,

CoLight predominantly focuses on local feature information, making it challenging to extract information about diverse scenarios.

5.7 Ablation Studies

To investigate the impact of each module, we conduct ablation experiments, including four settings: (1) without (w/o) the Co-Train module, (2) w/o the Experiential weight module, (3) w/o the Intersection-wise module, and (4) w/o the Feature-wise module.

Metrics Model	Delay	Trip Time	Wait
w/o Co-Train	-4.85%	-1.55%	-4.58%
w/o Experiential weight	-2.30%	-4.47%	-14.89%
w/o Intersection-wise weight	-5.53%	-1.61%	-6.71%
w/o Feature-wise weight	-1.57%	-2.85%	-4.66%

Table 3: The results of ablation experiments

Table 3 presents the results. Overall, the absence of any module will result in a decrease in model performance. Specifically, we observe that the Co-training and Intersection-wise weight are critical to improve delay, and the Experiential weight is essential to reduce the trip time and wait time. Full evaluation is in Table A4.

6 CROSS-SCENARIO NEIGHBORS

As mentioned before, our Co-Train + Experiential Weight design could easily be extended to even incorporate the "neighbors" from other scenarios. In this section, we provide some preliminary results and they are quite promising.

The illustration of how to select cross-scenario neighbors is shown in Fig. 6, elaborated as follows. Given an observation of a target intersection $o^{k,i}$, we first compute the cosine similarity S_c between the embedding of $o^{k,i}$ and all the observations from other scenarios $o^{k',\cdot}$, where $k' \in \{1, ..., K\} \backslash k$, as follows.

$$S_c(\Psi(o^{k,i}), \Psi(o^{k',\cdot})) := \frac{\Psi(o^{k,i}) \cdot \Psi(o^{k',\cdot})}{||\Psi(o^{k,i})|| \; ||\Psi(o^{k',\cdot})||} \qquad (15)$$

We design two ways of injecting cross-scenario neighbors. Way-(1) **the direct ones**: in Fig. 6(a), we select the messages of the top-k (here $k = 5$) correlated neighbors as the augmented external knowledge, or Way-(2) **the twin's**: in Fig. 6(b), we find the most similar neighbor (the twin) in scenario k' and we use the twin and its four neighbors, together with the four neighbors from the same scenario (then in total $N_{nei} = 9$), we enhance the decision-making process. Thus, benefiting from our framework design, we can directly get the weights from the intersection-wise weights with the least change: for Way-(1), we only need to re-train bigger intersection-wise weight matrice $weight_{int}^k \in \mathbb{R}^{N_k \times (1+N_{nei})}$ (here $1+N_{nei} = 10$), thus capturing the weight of all the N intersections in each scenario; for Way-(2), we can directly use the current weights $weight_{int}^{k,i} \in \mathbb{R}^5$ and $weight_{int}^{k',i'} \in \mathbb{R}^5$ to concat as a 10-dimensional weight to embed the 9 neighbors to $o^{k,i}$.

As a preliminary study, we follow Way-(1) only, with the initial intersection-wise parameter set as 1. Table 4, compared with our DuaLight, demonstrates that incorporating the neighbors across scenarios based on similarity can help the model improve performance even further.

(a) Directly choose cross-scenario neighbors

(b) Choose the twin intersections' neighbors
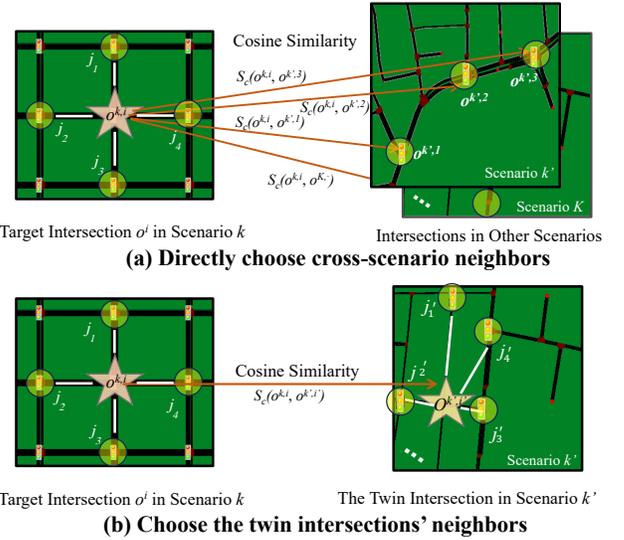
Figure 6: Two ways of picking cross-scenario neighbors

Metrics Scenarios	Delay	Trip Time	Wait
Grid 4 × 4	**48.59 ± 0.0**	**160.55 ± 0.0**	**23.02 ± 0.0**
Avenue 4 × 4	**693.82 ± 0.0**	530.03 ± 7.6	372.06 ± 0.0
Grid 5 × 5	**200.03 ± 0.0**	**217.46 ± 0.0**	**78.11 ± 0.0**
Cologne8	26.39 ± 0.0	90.71 ± 0.0	7.82 ± 0.0
Ingolstadt21	**168.1 ± 3.04**	**282.1 ± 8.7**	**95.87 ± 5.7**
Fenglin	**249.61 ± 0.0**	320.72 ± 0.0	171.02 ± 0.0
Nanshan	530.23 ± 0.0	696.28 ± 0.0	379.14 ± 0.0

Table 4: DuaLight++: with cross-scenario neighbors

7 CONCLUSION

This paper proposes a RL-based traffic signal control agent, with two integral components. The first is the Experiential Weighted module, which supports the model in learning dynamic information about both the neighboring feature weights and self-feature weights within a specific scenario. When combined with the GAT, these two weights empower the model to focus simultaneously on real-time neighbor information and environmental information inherent in the scenario. Secondly, we introduce the Co-train module, a component which is jointly trained with DQN across multiple scenarios. This facilitates the model's learning of shared and generic dynamic information across diverse scenarios. Our results suggest that DuaLight delivers SOTA performance, or performs competitively against the existing SOTA. Embedding's visualization reveals that DuaLight is capable of learning superior feature representations, enabling better decisions. Moreover, we give the promising result of incorporating neighbors from other scenarios.

Limitation and Future Work: The limitation is the requirement for retraining upon the addition of new scenarios. Future work will focus on developing a flexible weight learning mechanism to improve generalization to unseen scenarios, allowing for immediate adaptation without retraining.

REFERENCES

[1] David M Levinson. Speed and delay on signalized arterials. *Journal of Transportation Engineering*, 124(3):258–263, 1998.

[2] Alejandro Tirachini. Estimation of travel time and the benefits of upgrading the fare payment technology in urban bus services. *Transportation Research Part C: Emerging Technologies*, 30:239–256, 2013.

[3] The Economist. The cost of traffic jams. *The Economist: London, UK*, 2014.

[4] David Schrank, Bill Eisele, Tim Lomax, Jim Bak, et al. 2015 urban mobility scorecard. 2015.

[5] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*, 2019.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[7] Li Li, Yisheng Lv, and Fei-Yue Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.

[8] Xiaoyuan Liang, Xunsheng Du, Guiling Wang, and Zhu Han. Deep reinforcement learning for traffic light control in vehicular networks. *arXiv preprint arXiv:1803.11115*, 2018.

[9] Roger P Roess, Elena S Prassas, and William R McShane. *Traffic engineering.* Pearson/Prentice Hall, 2004.

[10] PB Hunt, DI Robertson, RD Bretherton, and M Cr Royle. The scoot on-line traffic signal optimisation technique. *Traffic Engineering & Control*, 23(4), 1982.

[11] PR Lowrie. Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. 1990.

[12] Carlos Gershenson. Self-organizing traffic lights. *arXiv preprint nlin/0411066*, 2004.

[13] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

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

[15] Elise Van der Pol and Frans A Oliehoek. Coordinated deep reinforcement learners for traffic light control. *Proceedings of learning, inference and control of multi-agent systems (at NIPS 2016)*, 8:21–38, 2016.

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

[17] Yanan Wang, Tong Xu, Xin Niu, Chang Tan, Enhong Chen, and Hui Xiong. Stmarl: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control. *IEEE Transactions on Mobile Computing*, 21(6):2228–2242, 2020.

[18] Libing Wu, Min Wang, Dan Wu, and Jia Wu. Dynstgat: Dynamic spatial-temporal graph attention network for traffic signal control. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2150–2159, 2021.

[19] Haoyuan Jiang, Ziyue Li, Lei Bai, Zhishuai Li, and Rui Zhao. A general scenario-agnostic reinforcement learning for traffic signal control, 2023.

[20] Huichu Zhang, Chang Liu, Weinan Zhang, Guanjie Zheng, and Yong Yu. Generalight: Improving environment generalization of traffic signal control via meta reinforcement learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1783–1792, 2020.

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

[22] Afshin Oroojlooy, Mohammadreza Nazari, Davood Hajinezhad, and Jorge Silva. Attendlight: Universal attention-based reinforcement learning model for traffic signal control. *Advances in Neural Information Processing Systems*, 33:4079–4090, 2020.

[23] Yican Lou, Jia Wu, and Yunchuan Ran. Meta-reinforcement learning for multiple traffic signals control. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4264–4268, 2022.

[24] LA Prashanth and Shalabh Bhatnagar. Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421, 2010.

[25] Xiaonan Klingbeil, Marius Wegener, Haibo Zhou, Florian Herrmann, and Jakob Andert. Centralized model-predictive cooperative and adaptive cruise control of automated vehicle platoons in urban traffic environments. *IET Intelligent Transport Systems*, 2023.

[26] Yiling Liu, Guiyang Luo, Quan Yuan, Jinglin Li, Lei Jin, Bo Chen, and Rui Pan. Gplight: grouped multi-agent reinforcement learning for large-scale traffic signal control. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 199–207, 2023.

[27] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[29] Shantian Yang and Bo Yang. A meta multi-agent reinforcement learning algorithm for multi-intersection traffic signal control. In *2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 18–25. IEEE, 2021.

[30] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[31] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421, 2020.

[32] Jinming Ma and Feng Wu. Feudal multi-agent deep reinforcement learning for traffic signal control. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 816–824, 2020.

[33] Christian Varschen and Peter Wagner. Mikroskopische modellierung der personenverkehrsnachfrage auf basis von zeitverwendungstagebüchern. *Integrierte Mikro-Simulation von Raum-und Verkehrsentwicklung. Theorie, Konzepte, Modelle, Praxis*, 81:63–69, 2006.

[34] Silas C Lobo, Stefan Neumeier, Evelio MG Fernandez, and Christian Facchi. Intas–the ingolstadt traffic scenario for sumo. *arXiv preprint arXiv:2011.11995*, 2020.

[35] James Ault and Guni Sharon. Reinforcement learning benchmarks for traffic signal control. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[36] Pravin Varaiya. The max-pressure controller for arbitrary networks of signalized intersections. *Advances in dynamic network modeling in complex transportation systems*, pages 27–66, 2013.

[37] Anastasios Kouvelas, Jennie Lioris, S Alireza Fayazi, and Pravin Varaiya. Maximum pressure controller for stabilizing queues in signalized arterial networks. *Transportation Research Record*, 2421(1):133–141, 2014.

[38] James Ault, Josiah P Hanna, and Guni Sharon. Learning an interpretable traffic signal control policy. *arXiv preprint arXiv:1912.11023*, 2019.

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

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

[41] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.