A Constrained Multi-Agent Reinforcement Learning Approach to Autonomous Traffic Signal Control

ANIRUDH SATHEESH, Computer Science, University of Maryland, College Park, United States
KEENAN POWELL, Computer Science, University of Maryland, College Park, United States

Traffic congestion in modern cities is exacerbated by the limitations of traditional fixed-time traffic signal systems, which fail to adapt to dynamic traffic patterns. Adaptive Traffic Signal Control (ATSC) algorithms have emerged as a solution by dynamically adjusting signal timing based on real-time traffic conditions. However, the main limitation of such methods is they are not transferable to environments under real-world constraints, such as balancing efficiency, minimizing collisions, and ensuring fairness across intersections. In this paper, we view the ATSC problem as a constrained multi-agent reinforcement learning (MARL) problem and propose a novel algorithm named Multi-Agent Proximal Policy Optimization with Lagrange Cost Estimator (MAPPO-LCE) to produce effective traffic signal control policies. Our approach integrates the Lagrange multipliers method to balance rewards and constraints, with a cost estimator for stable adjustment. We also introduce three novel constraints on the traffic network: GreenTime, GreenSkip, and PhaseSkip, which penalize traffic policies that do not conform to real-world scenarios. Our experimental results on three real-world datasets demonstrate that MAPPO-LCE outperforms three baseline MARL algorithms by across all environments and traffic constraints (improving on MAPPO by 12.60%, IPPO by 10.29%, and QTRAN by 13.10%). Our results show that constrained MARL is a valuable tool for traffic planners to deploy scalable and efficient ATSC methods in real-world traffic networks.

CCS Concepts: • **Computing methodologies** → **Multi-agent planning**; **Multi-agent reinforcement learning**; *Partially-observable Markov decision processes*; • **Mathematics of computing** → *Nonconvex optimization.*

Additional Key Words and Phrases: Multi-Agent, Traffic Signal Control, Reinforcement Learning, Constrained Optimization, Lagrange Multipliers

1 Introduction

Traditional traffic signal systems, which operate on pre-programmed, fixed schedules, are often inadequate in addressing the dynamic nature of urban traffic flow due to an inability to adapt to constantly changing traffic patterns. This can result in longer waiting times and unfair traffic distributions across intersections [11]. To combat the limitations of traditional fixed-time traffic signal systems, Adaptive Traffic Signal Control (ATSC) methods have been developed to adjust signal timing based on real-time traffic conditions dynamically. However, while ATSC methods hold promise in reducing congestion in busy intersections, there are still uncertainties about their deployment in real-world environments. One challenge is balancing efficiency while minimizing vehicle collisions and other hazards [15]. Another challenge is maximizing the fairness of each intersection, or ensuring that the green times (amount of time the current traffic light is green) for different lanes are the same on average [32]. In general, these challenges highlight the ongoing struggles with incorporating constraints into ATSC methods that accurately reflect the demands of real-world environments.

Authors' Contact Information: Anirudh Satheesh, Computer Science, University of Maryland, College Park, Maryland, United States; e-mail: anirudh.satheesh@gmail.com; Keenan Powell, Computer Science, University of Maryland, College Park, Maryland, United States; e-mail: kpowell1@terpmail.umd.edu.

ACM 2833-0528/2025/9-ART
https://doi.org/10.1145/3769308

Previous works on ATSC use the observations of the intersections to form traffic control policies, such as SOTL [10]. However, these are heuristic-based and cannot adapt to more complex traffic environments. Additionally, they do not consider how current actions can affect future states, which hinders long-term outcomes. Reinforcement Learning (RL) has also been used to develop autonomous traffic control methods by optimizing over current and future states [46]. This includes actor-critic methods [4] and policy gradient methods [27, 30] on single intersections [29] and multi-intersection environments [8, 42]. RL has also been used for non-traditional intersections such as roundabouts [34] and dynamical lane changing systems [48].

Due to the exponentially growing action space of reinforcement learning as the number of intersections increases, it becomes difficult to learn effective single-agent RL policies that can adapt to non-stationary environments like traffic signal control. As such, some works formulate ATSC as a decentralized Multi-Agent Reinforcement Learning (MARL) problem, using several agents to represent each intersection instead of one agent as a global traffic controller. This allows each intersection to act as its own local RL agent under partial observability and maximize its utility along with the global utility [38, 39, 48]. Additional work serves to improve baseline MARL algorithms by improving sample efficiency [20], or adding information to the state space to mitigate partial observability, such as communication methods [22] and environment modeling [5, 41].

Due to the efficacy of MARL in solving high-dimensional traffic control problems and current struggles with incorporating constraints that reflect real-world environments, we propose a constrained MARL algorithm named Multi-Agent Proximal Policy Optimization with Lagrange Cost Estimator (MAPPO-LCE). Specifically, the algorithm uses the Lagrange multipliers method to balance the constraints with maximizing rewards. MAPPO-LCE builds on existing constrained MARL algorithms such as MAPPO-Lagrange by introducing a cost estimator to alleviate the unstable policy updates when using the advantage function to update the Lagrange parameter.

Our contributions can be summarized as follows:

(1) We define three novel constraint functions: GreenSkip, GreenTime, and PhaseSkip, which penalize policies that do not reflect real-world scenarios.
(2) We propose a constrained MARL algorithm for multi-intersection traffic control and introduce a Lagrange Cost Estimator to alleviate potentially unstable constraint updates when using the advantage function.
(3) We show experimentally that MAPPO-LCE outperforms three baseline MARL algorithms on three different datasets.
(4) Our results show that constrained MARL can be a valuable tool for traffic planners to deploy ATSC methods in real-world traffic networks to reduce congestion.

2 Related Work

In this section, we discuss recent work on MARL algorithms and general constraints for ATSC.

2.1 MARL for ATSC

Recent work uses multi-agent reinforcement learning to model traffic signal control, with each agent controlling one intersection under partial observability. Wang et al. [39] developed independent and joint Advantage Actor-Critic (A2C) algorithms for ATSC with a centralized critic in a distributed setting. Chen et al. [9] also leverages A2C in a multi-agent setting, using decentralized critics for each agent in a distributed network. In addition to on-policy algorithms, previous works use multi-agent off-policy algorithms for ATSC. For example, Zhang et al. [45] uses Nash Q-Learning to alleviate the large state-action space from traditional MARL algorithms. Wang and Wang [38] improves on this by using a Deep Q-Network [26] with Friend Q-Learning [23] to achieve better coordination between agents.

Other ways to improve MARL algorithms in ATSC are to include additional information in each agent's observation space to create more informed policies. However, including more information does not always lead

to better results, as this can require more parameters and a slower convergence rate [47]. Thus, selecting the right information to include between agents is crucial for performance. Huang et al. [20] use a model-based approach by learning a global probabilistic dynamics model along with the policy, which generates a prediction of the next states as additional information. This method is purely decentralized, where there is no interaction between agents. Thus, Jiang et al. [22] develops UniComm, a method that computes only the necessary information between neighbor agents, which is used in their UniLight algorithm to calculate Q values for each agent.

2.2 Constraints for ATSC

Solving environments with incorporated constraints is difficult due to balancing rewards and costs from the constraints. Constrained Reinforcement Learning (CRL) is an active research area in RL that solves such environments by developing algorithms that exclusively learn policies that are both effective and satisfy the constraints (e.g. safety, fairness, etc.) [2, 17, 24]. Achiam et al. [2] develops a Constrained Policy Optimization (CPO) algorithm to learn policies under constraints, and Gu et al. [17] expands this into a multi-agent setting with MACPO and MAPPO-Lagrange. Tabas et al. [37] improve upon MACPO by developing a primal-dual optimization framework and parameterizing each agent with a neural network.

In ATSC, there is minimal work on incorporating constraints into the environment to develop policies closer to real-world scenarios. Gu et al. [16] partitions the traffic network topology to alleviate scalability issues with MARL, but this only constrains the state space, not the action space. Haydari et al. [18] use the CRL framework with the amount of emissions as the constraint and develop a Soft Actor-Critic algorithm to balance rewards with constraints. However, this is a single-agent setting, which poses scalability issues as the number of intersections increases. Adan et al. [3] models traffic environment constraints in a multi-agent setting, but this work models agents as the vehicles around one intersection, instead of each intersection being an agent. Finally, Raeis and Leon-Garcia [32] creates two fairness constraints for the ATSC problem, one delay-based metric which is meant to diminish the number of vehicles experiencing significantly longer waiting times and another throughput-based metric which attempts to give equal weighting to all traffic flows by extending concepts from computer networking. However, this is also a single agent setting in a more simplistic environment and is focused more specifically on fairness between the North-South and East-West traffic flows instead of constraints under general traffic network topologies.

3 Preliminaries

In this section, we define the Constrained Markov Game, the RL environment, and our constraints for ATSC.

3.1 Constrained Markov Game for ATSC

We can model ATSC as a constrained Markov Game [40] which can be represented by the tuple $M = \langle \mathcal{N}, S, \{O_i\}_{i\in\mathcal{N}}, \{A_i\}_{i\in\mathcal{N}}, \mathcal{T}, r, \Omega, C, c, \gamma\rangle$, where $\mathcal{N} = \{1, 2, ..., n\}$ is a set of n agents; S is the state space; $O = \times_{i\in\mathcal{N}} O_i$ is the joint observation space, where O_i is the observation space of agent i; $A = \times_{i\in\mathcal{N}} A_i$ is the joint action space, where A_i is the action space of agent i; $T : S \times A \times S \to [0, 1]$ is probabilistic state transition function; R is the reward function; $\Omega : S \times A \times O \to [0, 1]$ is space of conditional observation probabilities ($\Omega(s', a, o) = P(o|s', a)$); $C : S \times A \to \mathbb{R}$ is the cost function; and c is the cost limit. Since this is a decentralized Markov Game, the reward function for each agent is the same, e.g. $R = R_i \, \forall i \in \mathcal{N}$. MARL algorithms for constrained Markov Games aim to

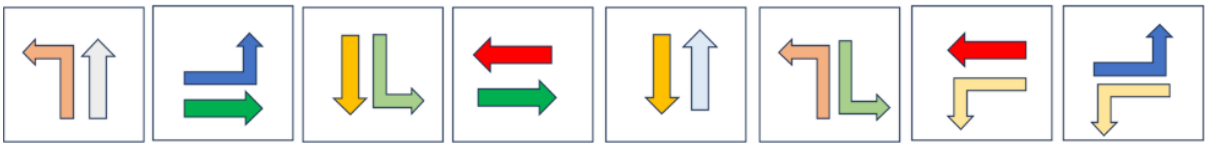

Fig. 1. 8 actions corresponding to the 8 phases for each intersection of traffic lights. Each phase corresponds to two traffic lights being on at the same time (e.g. the second box indicates vehicles are allowed to continue heading east or turn left to head north).

search for policy π that solves this constrained optimization problem:

$$\max_{\pi} \quad \mathbb{E}_{(s_t \sim S, a_t \sim \pi)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],$$

$$\text{s.t.} \quad \mathbb{E}_{(s_t \sim S, a_t \sim \pi)} \left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \right] \leq c$$

In the ATSC problem specifically, the elements of the environment are defined as:

- Agents: Each agent is responsible for controlling traffic lights at one intersection.
- Observation: The observation of each agent is composed of the characteristics of the corresponding intersection. Specifically, each intersection has 12 road links (vehicles turning left, right, and going straight in each cardinal direction), and each road link contains the number of vehicles moving, the number of vehicles waiting, the traffic light phase, and the number of vehicles in each lane, as well as the speed and location of each vehicle in the lane.
- Actions: As shown in Figure 1, there are eight phases that describe combinations of traffic lights that can be green simultaneously. At each timestep, the intersection can choose one of these phases as an action.
- STATE: The state is the combination of all observations at the current time step.
- STATE Transition: After an action is selected at each time step, vehicles are allowed to move if the corresponding traffic light is green for a short period T_g. While the environment does not directly represent yellow lights, before changing phases, all lights that would be turned on/off are turned to red for a brief period T_y before the lights of the new phase are turned to green.
- Reward: Each agent will receive a global reward $\lambda_f R_f + \lambda_w R_w$, where R_f is the total number of vehicles moving, R_w is the total number of vehicles waiting, and λ_f and λ_w are hyperparameters.

For more information on environment parameters, refer to Appendix A.

3.2 Environment Constraints

We develop three novel environment constraints on each intersection that reflect real-world environments named GreenTime, PhaseSkip, and GreenSkip. These constraints also help to promote fair treatment of all vehicles by the agents by reducing differences in waiting times between directions and encouraging agents to take all possible actions.

- GreenTime: Each light l should be green for no more than $G_{max\ time}$ before turning red to prevent long waiting times from other lanes, and model light cycles in the real world. Each time step that a light is on increases its GreenTime value by 1, and when it is turned off its GreenTime value is constant at 0. This ensures that no specific lane is green for an unrealistically large amount of time. Right-turn lights are

ignored for this constraint, as they are always treated as being on.

$$G_{time}(l) \leq G_{max\ time} \tag{1}$$

- PhaseSkip: The state of each traffic light follows one of a specific, pre-determined set of phases (see Figure 1). No phase should be skipped consecutively more than $P_{max\ skip}$ times. Each time the phase changes, the new phase has its PhaseSkip value set to 0, and all phases other than the new phase and the old phase have their PhaseSkip values incremented by 1. This is a way of somewhat closely approximating how traffic cycles work in the real world, by rotating roughly evenly between possible phases, as well as being an indirect way of promoting the agent to give equal attention to all lanes.

$$P_{skips}(p) \leq P_{max\ skips} \tag{2}$$

- GreenSkip: Similar to the phase constraint, no individual light l should be skipped consecutively more than $G_{max\ skips}$ times. Each time the phase changes, each light turned on in the new phase has its GreenSkip value set to 0, and all lights not on in the new phase or the old phase have their GreenSkip values incremented by 1. This is a direct way of promoting fairness by reducing the variance in waiting times among all lanes, as this ensures that if the phase is continuously changing, each lane will receive some amount of attention within the phase cycle. Right-turn lights are also ignored for this constraint.

$$G_{skips}(l) \leq G_{max\ skips} \tag{3}$$

Each agent is constrained according to Eqns 1-3. The penalty associated with each constraint is the average across all lights:

$$\frac{\sum_{i \in \mathcal{N}} \sum_{l} \frac{1_c}{n_l(i)}}{|\mathcal{N}|} \tag{4}$$

where 1_c is an indicator function that checks whether the constraint is satisfied, i is the intersection, $|\mathcal{N}|$ is the number of agents, l is a specific light at the intersection the agent controls, and $n_l(i)$ is the total number of lights at the intersection that particular agent controls. Note that for the PhaseSkip constraint, we sum over the phases and divide by the total number of phases. For our experiments, the number of lights is always 12 and the number of phases is always equal to 8, as all the intersections in our environment have 4 roads of 3 lanes with 8 distinct phases. The exact algorithms for calculating each constraint can be referenced in Algorithms 1, 2, and 3, and examples for each constraint can be seen in Figures 2, 3, 4.

3.3 Comparison to Alternate Fairness Definitions

Our fairness constraints offer two key advantages over alternative definitions commonly used in multi-agent reinforcement learning (MARL): they are more interpretable and can be naturally integrated into the constrained MARL framework, which is significantly easier to optimize. Two widely used approaches for incorporating fairness into reinforcement learning are Generalized Gini Functions (GGFs) [35] and the Coefficient of Variation [21].

GGFs work by defining a sorted D-dimensional vector of agent utilities v and a corresponding sorted weight vector w, then computing their dot product. This presents two challenges in comparison to our approach. First, managing how the sorted objective vector v evolves as policies are updated becomes increasingly complex as the number of agents grows. Second, GGFs lack the intuitive interpretability of our constraints. For instance, when using GreenTime as a constraint in traffic networks, traffic planners can directly observe the duration of green lights and how constraint violations decrease over time (see Section 6.2), making the fairness behavior more transparent.

The Coefficient of Variation, another alternative, measures the variation of agents' utilities around their mean. While this is an interpretable metric, it is difficult to optimize because it is embedded directly into the reward

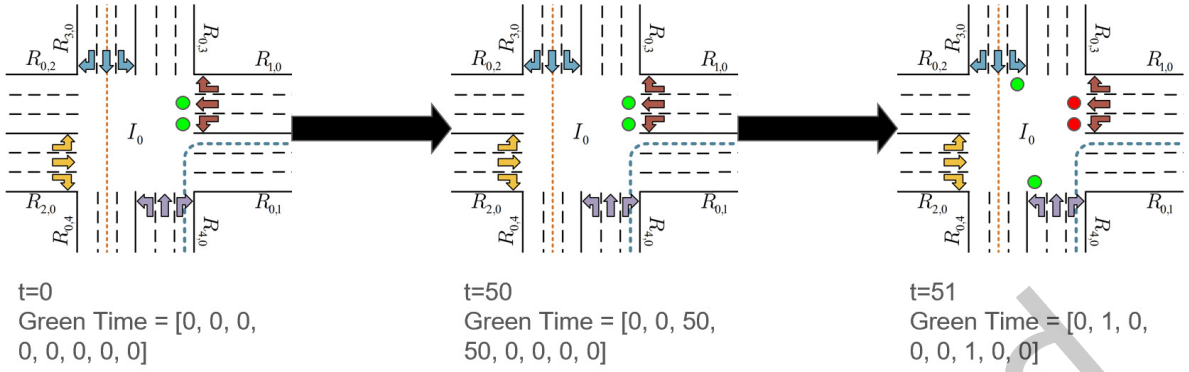

Fig. 2. The figure above shows an example of the GreenTime constraint, with time shown as t=, and the array being the constraint value for each light in the example intersection. As explained above, each lights GreenTime value is simply the time it has been on in a row, with larger values being penalized.

function and thus requires coordination through a joint policy across all agents. In contrast, our method decouples constraint dynamics from the primary objective by introducing a separate MDP to manage constraint learning. This separation avoids the scalability and optimization issues that joint reward-based fairness formulations have.

Algorithm 1 GreenTime Calculation

```
for time = 1 to N do
    for light in lights do
        if light is ON in the current phase then
            green_time[light] ← green_time[light] + 1
        else
            green_time[light] ← 0
        end if
    end for
end for
```

Algorithm 2 PhaseSkip Calculation

```
for time = 1 to N do
    if new_phase ≠ old_phase then
        for phase in phases do
            if (phase ≠ old_phase) and (phase ≠ new_phase) then
                phase_skips[phase] ← phase_skips[phase] + 1
            end if
        end for
        phase_skips[new_phase] ← 0
    end if
end for
```

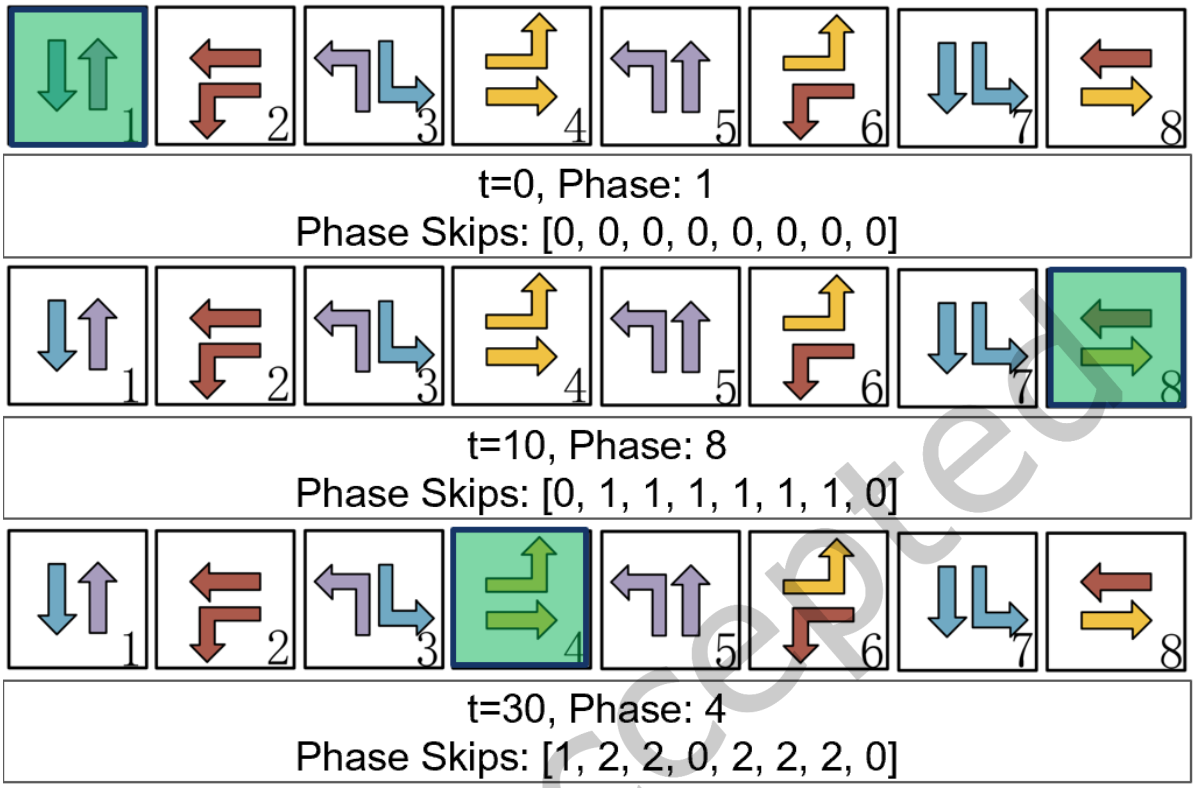

Fig. 3. The figure above shows an example of the PhaseSkip constraint, with time shown as t=, and the array being the constraint value for each phase in the example intersection. As explained above, each phase's PhaseSkip value is simply the number of times it has been passed over in phase changes without being selected or being the previous phase, with larger values being penalized.

Algorithm 3 GreenSkip Calculation

```
for time = 1 to N do
  if new_phase ≠ old_phase then
    for light in lights do
      if (light is RED in old_phase) and (RED in new_phase) then
        green_skips[light] ← green_skips[light] + 1
      else
        green_skips[light] ← 0
      end if
    end for
  end if
end for
```

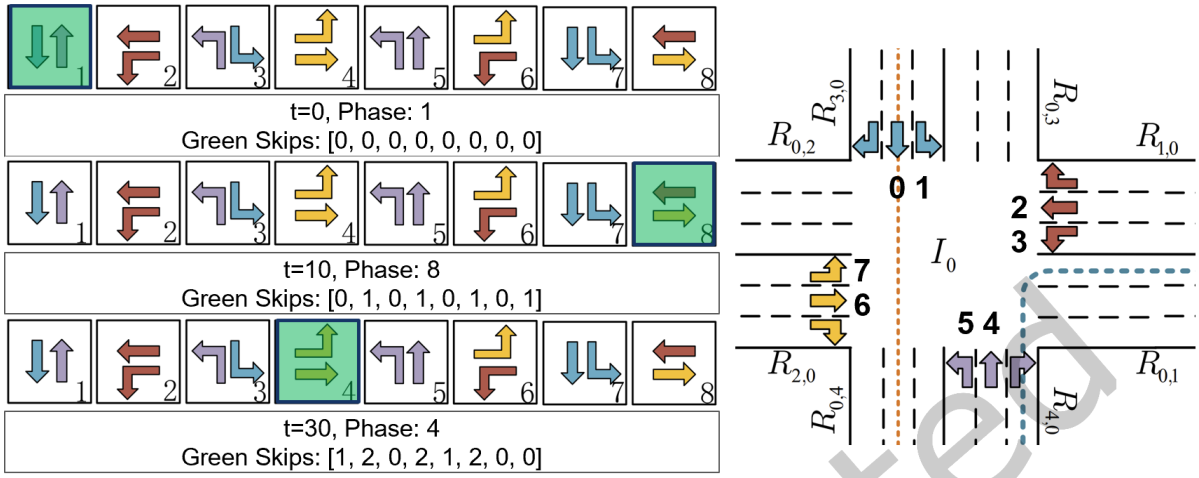

Fig. 4. The figure above shows an example of the GreenSkip constraint at time t, and the array being the constraint value for each light in the example intersection. As explained above, each light's GreenSkip value is simply the number of times it has been passed over in phase changes without being selected or being on in the previous phase, with larger values being penalized.

4 Method

In this section, we describe our constrained multi-agent reinforcement learning algorithm: Multi-Agent Proximal Policy Optimization with Lagrange Cost Estimator (MAPPO-LCE).

4.1 Multi-Agent Proximal Policy Optimization with Lagrange Cost Estimator

Constrained optimization problems are typically of the form

$$\max_x f(x)$$
$$\text{s.t. } g(x) \leq c$$

which can be solved by the Lagrange multiplier method

$$\mathcal{L}(x;\lambda) = f(x) - \lambda(g(x) - c) \tag{5}$$

where $\mathcal{L}(x;\lambda)$ is a new optimization objective to maximize and $\lambda > 0$ is the Lagrange multiplier. [6]. Thus, for the constrained MARL problem,

$$\max_{\pi_\theta} \ \mathbb{E}_{(s_t \sim S, a_t \sim \pi_\theta)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \tag{6}$$

$$\text{s.t.} \quad \mathbb{E}_{(s_t \sim S, a_t \sim \pi_\theta)} \left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \right] < c \tag{7}$$

we can formulate it with a Lagrangian where

$$f(x) = \mathbb{E}_{(s_t \sim S, a_t \sim \pi)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \tag{8}$$

$$g(x) = \mathbb{E}_{(s_t \sim S, a_t \sim \pi)} \left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \right] \tag{9}$$

In MAPPO-LCE, we use a reward critic and a cost critic, $V^r_{\phi_r}$ and $V^c_{\phi_c}$, for estimating the discounted cumulative reward and discounted cumulative cost, respectively. We choose to build off of MAPPO because we require only one actor and one critic model during training and inference, which reduces the computation and memory requirements of the algorithm. Instead of training on every step, we also collect a dataset D containing rollout data every episode: $\{s_t, r_t, c_t, s_{t+1}\}$. After B episodes, we update the policy. Similar to MAPPO-Lagrange [17], we aim to minimize the following loss:

$$\mathcal{L}(\pi_\theta) = \mathcal{L}_r(\pi_\theta) - \lambda \mathcal{L}_c(\pi_\theta) \tag{10}$$

where $\mathcal{L}_r$ and $\mathcal{L}_c$ are the MAPPO [43] actor losses with an unclipped critic loss term:

$$\begin{aligned} \mathcal{L}_r(\pi_\theta) =& \mathbb{E}_{s_t \sim D, a_t \sim \pi_\theta} \left[\min \left(\rho_t A^r_t, \text{clip}(\rho_t, 1 \pm \epsilon) A^r_t \right) \right] \\ &+ \beta \mathbb{E}_{s_t \sim D} \left[\frac{1}{2} \| V^r_{\phi_r}(s_t) - r_t \|^2 \right] \end{aligned} \tag{11}$$

$$\begin{aligned} \mathcal{L}_c(\pi_\theta) =& \mathbb{E}_{s_t \sim D, a_t \sim \pi_\theta} \left[\min \left(\rho_t A^c_t, \text{clip}(\rho_t, 1 \pm \epsilon) A^c_t \right) \right] \\ &+ \beta \mathbb{E}_{s_t \sim D} \left[\frac{1}{2} \| V^c_{\phi_c}(s_t) - c_t \|^2 \right] \end{aligned} \tag{12}$$

In these formulations, ρ_t is the importance sampling ratio

$$\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

A^c_t and A^r_t are the cost advantage and reward advantage functions respectively, and ϵ is the clipping parameter. Here, we abuse notation and say that $A^c_t = A^c_t(s_t, a_t)$ and $A^r_t = A^r_t(s_t, a_t)$. We also update the reward critic model $V^r_{\phi_r}$ and the cost critic model $V^c_{\phi_c}$ by their respective temporal difference error (TDE):

$$\mathcal{L}_{\phi_r} = \mathbb{E}_{(s_t, s_{t+1}) \sim D} \left[r_t + \gamma V^r_{\phi_r}(s_{t+1}) - V^r_{\phi_r}(s_t) \right] \tag{13}$$

$$\mathcal{L}_{\phi_c} = \mathbb{E}_{(s_t, s_{t+1}) \sim D} \left[c_t + \gamma V^c_{\phi_c}(s_{t+1}) - V^c_{\phi_c}(s_t) \right] \tag{14}$$

In MAPPO-Lagrange [17], the Lagrange multiplier λ is updated by the mean of the cost advantage function A^c_t. This works in theory because the cost advantage function measures how much constraint violation occurs in a certain state when taking a particular action, compared to the mean constraint violation over all actions. Thus, if taking an action in a state results in a negative cost advantage, λ should be increased to alleviate this. However, since the advantage function only converges during the policy learning process, it may take many iterations to accurately estimate the constraint violation. During this time, the estimates can be unstable and potentially incorrect. To address this issue, we incorporate a Lagrange Cost Estimator to provide more stable and reliable estimates of constraint violations. This cost estimator quickly learns the cost dynamics within the first

few iterations to accurately predict the cost, and then updates λ. We train the cost estimator θ_C by minimizing the following loss:

$$\mathcal{L}_{\theta_C} = \|\theta_C(s_t, a_t) - c_t\|^2,\ s_t \sim D, a_t \sim \pi_\theta \tag{15}$$

Finally, we update λ with the following loss to ensure that the constraint function is satisfied under the cost limit c:

$$\mathcal{L}_\lambda = \mathbb{E}_{s_t \sim D, a_t \sim \pi_\theta}\left[-\lambda(\theta_C(s_t, a_t) - c)\right] \tag{16}$$

as the loss is minimized when the estimated cost is much less than the cost limit. One consideration is that instead of updating the policy in a fully online manner, we perform rollouts of the MARL policy for one episode and store the trajectories (containing the state, action, reward, cost, and next state) in a replay buffer. Then during training, we randomly sample from this replay buffer. The main advantage is that each agent learns from a more diverse set of environment updates and reduces the variance of gradient updates. Additionally, we clamp λ to be greater than zero to ensure that the policy is always penalized when the constraints are violated. Finally, to allow for smoother transitions between updates in the actor model and the critic models, we perform soft updates using the frozen versions of the models used in the Temporal Difference Error calculations. We display the full algorithm in Algorithm 4.

Algorithm 4 MAPPO-LCE Algorithm

Initialize replay buffer $\mathcal{D}$, policy parameters θ, critic networks V_ϕ^r, V_ϕ^c, cost network θ_C, and Lagrange multiplier λ.
for each episode **do**
  for each time step t **do**
    Select action $a_t = \pi_\theta(s_t)$
    Execute joint action a_t at state s_t
    Observe reward r_t, cost c_t, and next state s_{t+1}
    $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r_t, c_t, s_{t+1})$
  end for
  Sample batch $\mathcal{B}$ from $\mathcal{D}$
  $\theta \leftarrow \theta - \alpha\nabla\mathcal{L}(\pi_\theta)$ by Equation 10
  $\phi^r \leftarrow \phi^r - \alpha\nabla\mathcal{L}_{\phi_r}$ by Equation 13
  $\phi^c \leftarrow \phi^c - \alpha\nabla\mathcal{L}_{\phi_c}$ by Equation 14
  $\theta_C \leftarrow \theta_C - \alpha_{\theta_C}\nabla\mathcal{L}_{\theta_C}$ by Equation 15
  $\lambda \leftarrow \lambda - \alpha_\lambda\nabla_\lambda\mathcal{L}_\lambda$ by Equation 16
  Clamp to ensure $\lambda \geq 0$
  Soft update actor and critic parameters:

$$\theta \leftarrow \tau\theta + (1-\tau)\theta', \quad \phi_r \leftarrow \tau\phi_r + (1-\tau)\phi_r', \quad \phi_c \leftarrow \tau\phi_c + (1-\tau)\phi_c'$$

end for

5 Experiments

In this section, we outline our experimental details, including the environment configurations and explanation of baseline algorithms.

5.1 Environment Setup

We run our experiments on MAPPO-LCE and related baselines on the CityFlow environment [44], which is a scalable and realistic traffic simulator due to its C++ backend. Additionally, it is compatible with several multi-agent RL algorithms by integrating with the Gymnasium library [7]. From Wei et al. [42], there are three publicly available datasets collected from real-world traffic data from Hangzhou, China (HZ); Jinan, China (JN); and New York, USA (NY). Details of each environment are located in Table 1.

Each environment is defined by a fixed network file, which outlines the topology of the traffic network and positions the traffic signals and roads as coordinates in space. There is a also a set traffic flow file, which defines the route each vehicle will take. The difficulty with each environment is that the vehicles will randomly enter the simulation, with varying distributions on where and when vehicles will appear. As such, a successful policy must be able to adapt to a wide range of traffic flow distributions.

To evaluate the performance of each of the MARL algorithms, we use three evaluation metrics:

- Test Reward: The test reward is the same as the training reward: $\lambda_f R_f + \lambda_w R_w$.
- Average Delay: The average delay is the average delay across all vehicles, which is the total travel time minus the expected travel time for each vehicle. The expected travel time is the estimated time the vehicle should finish its route if there were no traffic lights.
- Throughput: The throughput of the environment is the number of vehicles that complete their routes before the episode ends.

	HZ	JN	NY
Number of Intersections	16	12	48
Number of Lanes	3	3	3
Total Number of Vehicles	2983	6295	2824
Time Steps (s)	3600	3600	3600

Table 1. Summary of Traffic Metrics for HZ, JN, and NY.

5.2 Baseline Methods

In this work, we compare our algorithm to three baseline MARL algorithms: Independent Proximal Policy Optimization (IPPO) [13], Multi-Agent Proximal Policy Optimization (MAPPO) [43], and QTRAN [36]. This set of algorithms allows us to test both on-policy algorithms (IPPO, MAPPO), and off-policy algorithms (QTRAN).

- IPPO [13]: IPPO treats each agent as its independent local RL agent to maximize local rewards. This transforms the problem into $|\mathcal{N}|$ independent single-agent PPO rollouts.
- MAPPO [43]: MAPPO joins the actions of each agent into a single joint action vector, and each agent shares an actor network and a critic network to update the policy.
- QTRAN [36]: QTRAN develops an unstructured value function factorization, which allows for more generalizable decentralized execution of MARL problems.

All baseline algorithms were implemented or derived from the ePYMARL library [31]. For each algorithm, the total reward at time step t is $r_t - \zeta c_t$, where r_t and c_t are the rewards and costs at time step t, and ζ is a hyperparameter that trades off maximizing the reward and satisfying the constraints. All experiments were conducted on a single RTX A5000 GPU.

6 Results

In this section, we show the results of our algorithm on several environment configurations and perform ablation studies to highlight the advantages of specific components of MAPPO-LCE.

6.1 Main Results

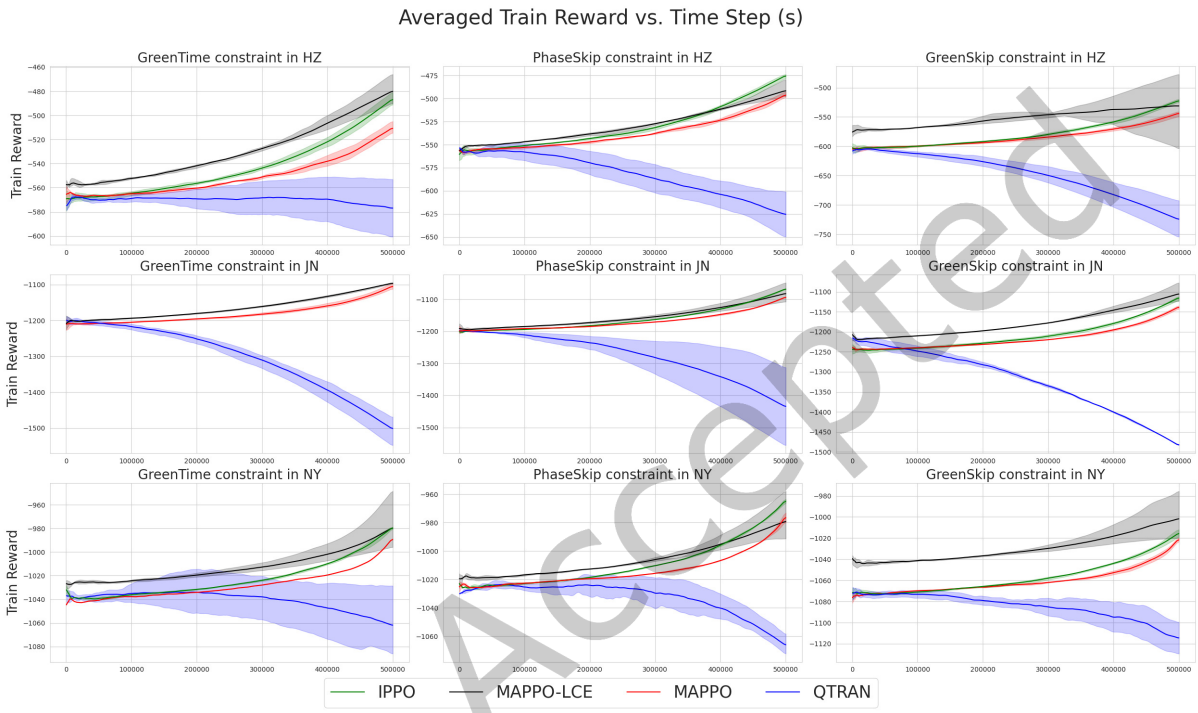

Fig. 5. Plot of train reward on the over 500,000 timesteps for the MAPPO-LCE algorithm compared to baseline algorithms across all environments and constraints. Plots are best viewed in color.

The results of the algorithms on the three environments are shown in Figure 10 and Table 2. In these figures, we include the results of each algorithm on the test reward function defined in Section 5.1.

As shown in Table 2, MAPPO-LCE outperforms all three comparison algorithms in every combination of environment and constraint that was tested on. While some of the other algorithms come close to the performance of MAPPO-LCE on specific setups (e.g. IPPO on HZ GreenSkip or QTRAN on NY PhaseSkip), taking the average across all runs yields a 12.60% improvement over MAPPO, a 10.29% improvement over IPPO, and a 13.10% improvement over QTRAN. Additionally, taking the average over the different constraints, MAPPO-LCE sees a 13.05% improvement with GreenTime, a 12.08% improvement with PhaseSkip, and a 10.87% improvement with GreenSkip. The slight decay in improvement with PhaseSkip or GreenSkip is likely due to them over-constraining the action space and too strongly encouraging the model to switch into unoptimal phases too often, but even with those restrictions, the model still sees consistent improvements.

To demonstrate the improved sample efficiency of MAPPO-LCE compared to existing MARL algorithms, we show the reward on the training environment over time in Figure 5. MAPPO-LCE demonstrates better sample

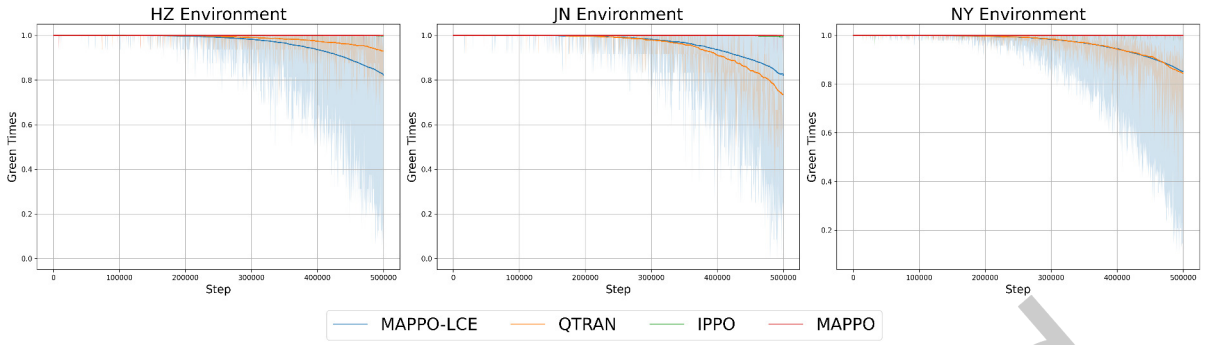

Fig. 6. Plot of train Green Time constraint values over all environments and algorithms. Plots are best viewed in color.

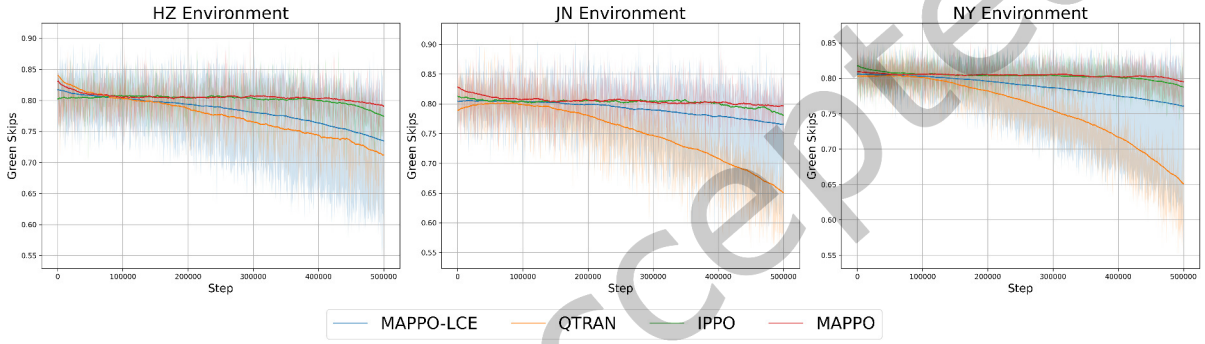

Fig. 7. Plot of train Green Skip constraint values over all environments and algorithms. Plots are best viewed in color.

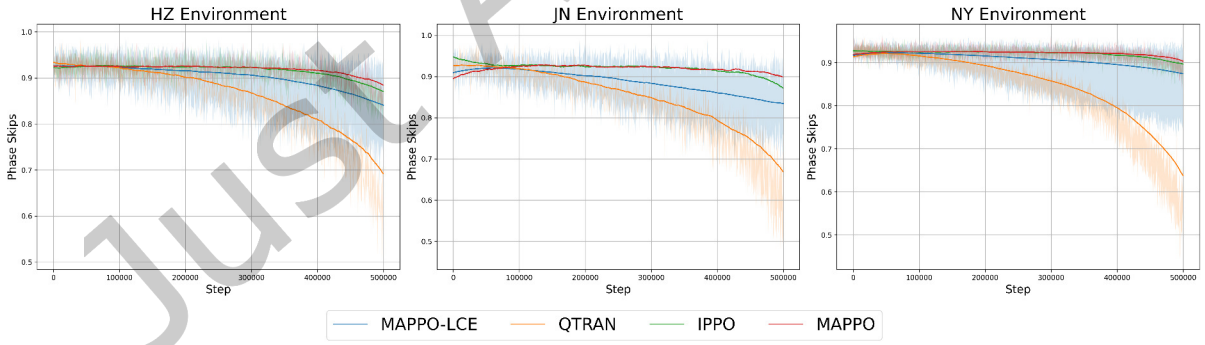

Fig. 8. Plot of train Phase Skip constraint values over all environments and algorithms. Plots are best viewed in color.

efficiency in all environment and constraint combinations. Although in some environments MAPPO and IPPO obtain higher rewards, this is at the cost of higher constraint violation rates, as shown in Figures 6, 7, and 8. Interestingly, we note the opposite conclusion for QTRAN, which has a significant decrease in reward over training but also decreased constraint violation rates. Unlike MAPPO and IPPO, QTRAN emphasizes reducing the

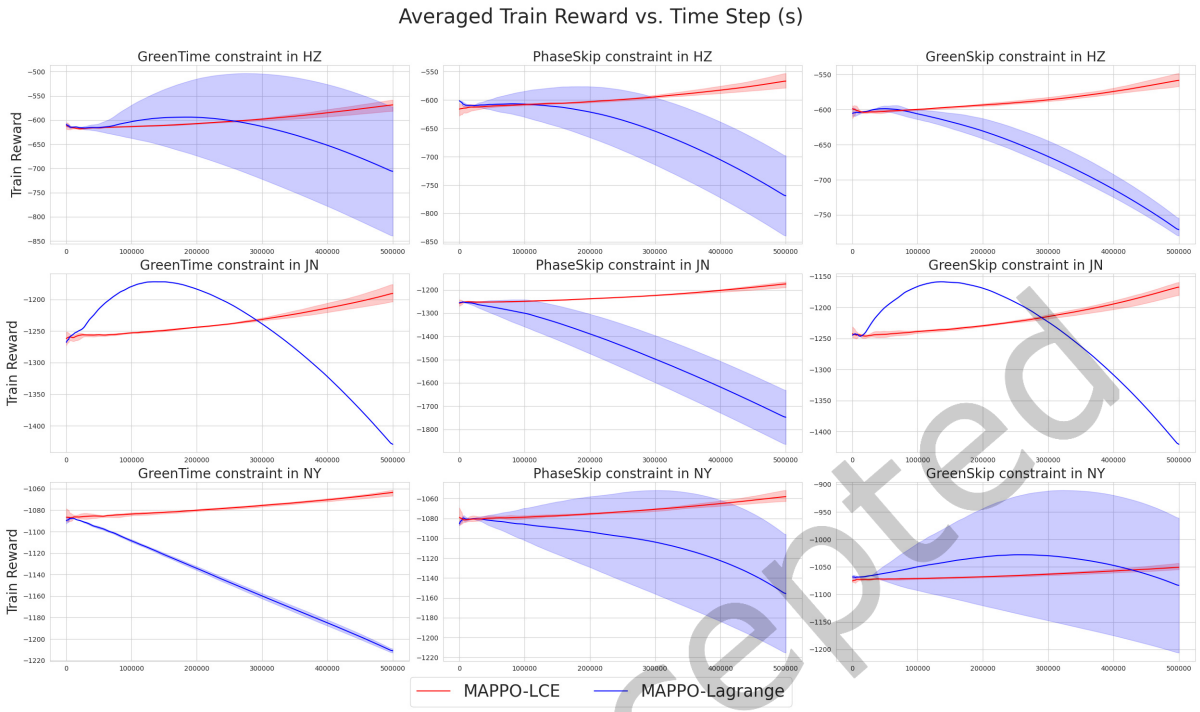

Fig. 9. Plot of train reward on the over 500,000 timesteps for the MAPPO-LCE algorithm compared to MAPPO-Lagrange across all environments and constraints. Plots are best viewed in color.

Environment	Constraint	MAPPO-LCE % Reward increase over comparison algorithms		
		MAPPO	IPPO	QTRAN
HZ	GreenTime	**13.86%**	12.15%	10.61%
HZ	PhaseSkip	**12.88%**	10.99%	8.74%
HZ	GreenSkip	11.27%	2.55%	**21.58%**
JN	GreenTime	14.22%	7.65%	**21.74%**
JN	PhaseSkip	**18.57%**	15.03%	17.95%
JN	GreenSkip	**10.69%**	7.73%	7.39%
NY	GreenTime	10.4%	**14.15%**	12.62%
NY	PhaseSkip	9.46%	**9.52%**	5.55%
NY	GreenSkip	12.05%	**12.83%**	11.75%

Table 2. Comparison of the Test Reward metric between MAPPO-LCE and MARL baseline algorithms across all constraint and environment combinations.

constraint values, but to a point that decreases the utility of the algorithm. These results show that MAPPO-LCE is able to accurately manage balancing constraints and generating an effective MARL policy.

Environment	Constraint	MAPPO-LCE % Throughput increase over comparison algorithms		
		MAPPO	IPPO	QTRAN
HZ	GreenTime	**11.4%**	-5.93%	-0.59%
HZ	PhaseSkip	**10.66%**	-5.44%	1.81%
HZ	GreenSkip	11.88%	-13.73%	**34.89%**
JN	GreenTime	23.9%	10.45%	**57.99%**
JN	PhaseSkip	29.77%	11.3%	**29.86%**
JN	GreenSkip	**15.42%**	0.46%	11.85%
NY	GreenTime	40.25%	7.43%	**50.88%**
NY	PhaseSkip	**78.22%**	18.86%	44.02%
NY	GreenSkip	**74.15%**	15.6%	63.91%

Table 3. Comparison of the Test Throughput metric between MAPPO-LCE and MARL baseline algorithms across all constraint and environment combinations. Here, a higher metric indicates a better policy.

Environment	Constraint	MAPPO-LCE % Average Delay decrease		
		MAPPO	IPPO	QTRAN
HZ	GreenTime	24.23%	8.56%	**20.44%**
HZ	PhaseSkip	**13.73%**	-2.97%	13.31%
HZ	GreenSkip	10.83%	-17.89%	**24.72%**
JN	GreenTime	15.64%	8.19%	**24.44%**
JN	PhaseSkip	19.27%	10.52%	**19.78%**
JN	GreenSkip	**10.6%**	2.11%	8.51%
NY	GreenTime	8.36%	3.94%	**10.47%**
NY	PhaseSkip	**13.41%**	6.71%	11.52%
NY	GreenSkip	**11.65%**	4.79%	11.94%

Table 4. Comparison of the Test Average Delay metric between MAPPO-LCE and MARL baseline algorithms across all constraint and environment combinations. Note that a higher metric here is better due to an average delay decrease.

One additional notable result from Figure 10 is that our algorithm more consistently outperforms the comparison algorithms as the complexity of the environment increases (we consider the NY environment the most complex, as it contains 3-4 times more agents than the HZ and JN environments). This is because in harder environments with more agents, policies are more heavily penalized for violating constraints. MAPPO-LCE's ability to adaptively handle constraints through a learnable Lagrange multipliers parameter enables it to more effectively balance performance while adhering to constraints across all agents. This underscores the scalability of MAPPO-LCE, which is essential for incorporating MARL policies in real-world settings.

A further analysis of the results can be done by looking at Tables 3 and 4, which show other important metrics for traffic systems (Figures 11 and 12). MAPPO-LCE is sometimes outperformed in either or both of these metrics by other algorithms - however, this only occurs in the HZ environment, which is the simplest. MAPPO-LCE still outperforms all three comparison algorithms in general, however, it has much smaller margins of improvement over IPPO. In terms of Throughput, it achieves a 32.85% improvement over MAPPO, a 32.73% improvement over QTRAN, but only a 4.33% improvement over IPPO. In terms of Average Delay, it achieves a 14.19% improvement

over MAPPO, a 16.13% improvement over QTRAN, but only a 2.66% improvement over IPPO. This is likely because the traffic environment is largely independent, as while nearby traffic lights exhibit some dependencies due to their close proximity, the influence of distant traffic lights is minimal and delayed. Thus, an independent policy algorithm like IPPO can effectively capture localized decision-making dynamics while avoiding unnecessary coordination. Additionally, De Witt et al. [13] shows that IPPO's performance on fully cooperative tasks can exceed other on-policy algorithms like MAPPO and Q-function factorization methods such as QMIX [33] and QTRAN due to the importance of policy value clipping. However, IPPO still lags behind MAPPO-LCE due to the Lagrange Cost Estimator's ability to guide the policy in the implicitly constrained state and action space.

We also show the averaged individual test constraint values for each of the algorithms in Figures 13, 14, and 15, where each constraint value of a particular constraint ranges from 0-1 representing the proportion of lights/phases in violation of the constraint, with 0 being no violations and 1 being all possible violations. In almost all test environment and constraint combinations, the value of the constraint violation is lowest for MAPPO-LCE compared to the other baseline algorithms. Additionally, QTRAN and MAPPO's performance decrease is supported by the high constraint violations, especially for GreenSkip and PhaseSkip, as the number of constraint violations increases significantly during training. However, while IPPO has much better performance in each environment than QTRAN and MAPPO (but not as much as MAPPO-LCE), it still has large constraint violations, especially concerning the GreenSkip constraint. Finally, we note that MAPPO-LCE's improved performance on the harder JN and NY datasets is supported by the lower constraint values for those two datasets, which continues to demonstrate the algorithm's scalability.

We also compare our algorithm to MAPPO-Lagrange [17] in Figure 9 on all environments and constraints. Figure 9 clearly illustrates the divergent behaviors of the two algorithms, as MAPPO-LCE steadily improves the training reward linearly, whereas MAPPO-Lagrange initially rises but begins to decline sharply after a few thousand steps. In MAPPO-Lagrange, λ is updated by the raw cost advantages calculated by Monte Carlo returns, which introduces high variance and non-stationarity. Over time, this results in a policy that shifts from pursuing reward to over-constraining, causing rapid performance degradation. By contrast, MAPPO-LCE maintains an increasing reward curve due to its learned cost estimator, which converges much faster since it is decoupled from the policy, which allows the policy to focus on updating $V^r_{\phi_r}$ and $V^c_{\phi_c}$ more effectively instead of overoptimizing λ.

6.2 Further Results

In Figures 11 and 12, we show the throughput and average delay results on the test set across timesteps for the combinations of environments and constraints. Additionally, we provide experiment results for the setting where all constraints are combined linearly to test each algorithm's ability to handle multiple constraints in Tables 5, 6, and 7. When evaluating this challenging setting where all constraints are combined, MAPPO-LCE continues to achieve the highest overall reward, outperforming MAPPO, IPPO, and QTRAN by 6.43%, 0.22%, and 4.13%, respectively (see Tables 5, 6, and 7). While the performance margins are smaller than in individual constraint settings, this is expected due to the inherent complexity of optimizing conflicting objectives (GreenTime, PhaseSkip, and GreenSkip simultaneously). Still, even in this more difficult scenario, MAPPO-LCE remains the most robust algorithm, demonstrating its ability to learn balanced policies under multiple constraints. Interestingly, IPPO occasionally achieves marginally better throughput (2.25%) and delay (0.58%) than MAPPO-LCE in this setting, likely due to the increased optimization instability introduced by competing constraints. However, IPPO's gains come at the cost of higher constraint violations (see Figure 16). Figure 16 shows the summed constraint values across the test runs on each environment. Notice that while MAPPO-LCE only attains a slightly lower amount of constraint violations compared to IPPO on HZ (with IPPO trending lower at the end), it by far trends the lowest on both of the more complex JN and NY datasets. This again highlights MAPPO-LCE's ability to

reliably enforce constraints without sacrificing quality reward optimization, as it maintains competitive or better metrics on reward, throughput, and delay, while much more strongly minimizing constraint violations.

To further improve the effectiveness of MAPPO-LCE when there are multiple constraints, we can introduce multiple Lagrange multipliers to measure each constraint individually instead of their aggregate. This should result in a much more stabler cost estimator learning during training because the individual constraint dynamics are easier to model, culminating in a policy that generalizes well to many constraints. However, introducing more Lagrange multipliers necessitates more cost estimators, each with their separate optimizers, which could increase the overhead of the algorithm. One potential way to circumvent this overhead is to use a shared backbone and separate prediction heads for each constraint, but we leave this to future work.

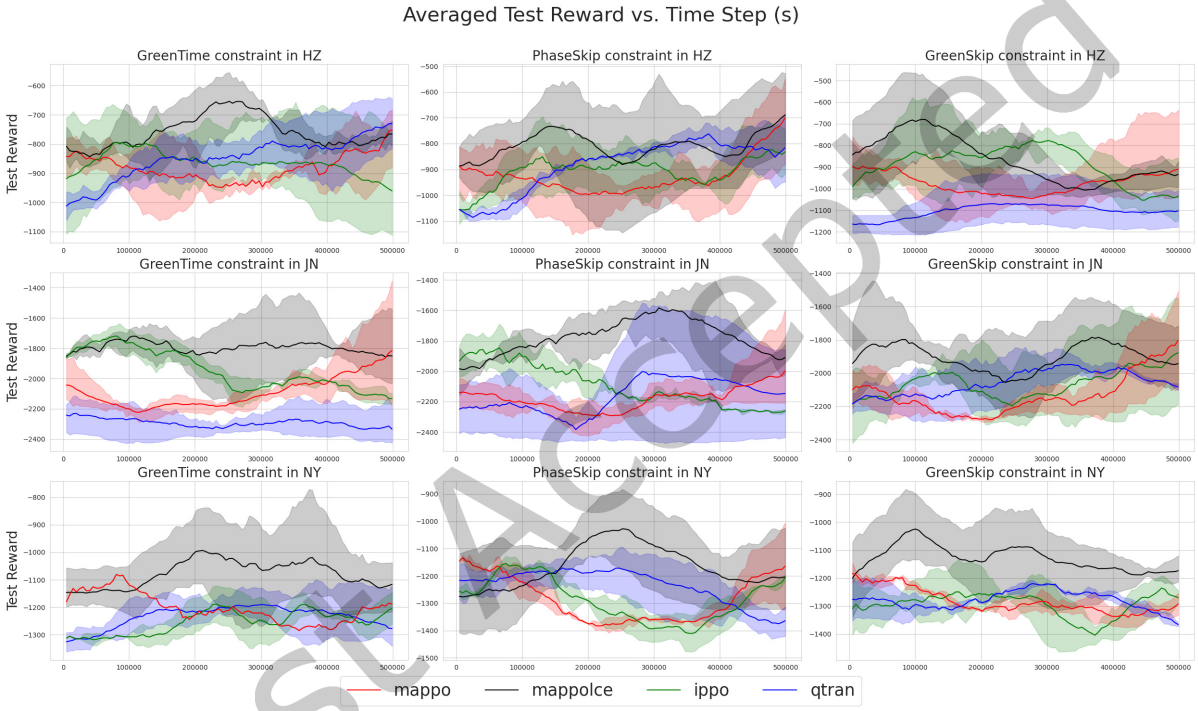

Fig. 10. Plot of test reward on the over 500,000 timesteps for the MAPPO-LCE algorithm compared to baseline algorithms across all environments and constraints. Plots are best viewed in color.

7 Future Work

For future work, one idea is to incorporate communication between agents, as such communication is already implemented in practice via connected vehicle technology [1]. This would allow traffic lights to communicate information such as the number of vehicles traveling from one intersection to another, the average travel time, and other vital data that could help discover more optimal policies. This also can be easily represented with an underlying Graph Neural Network (GNN), which means we can add message loss to the overall loss as both the GNN and the neural networks in each algorithm use backpropagation. Additionally, following from Section 6.1, excessive communication may be unnecessary and potentially prohibitive. Thus, leveraging pruning strategies

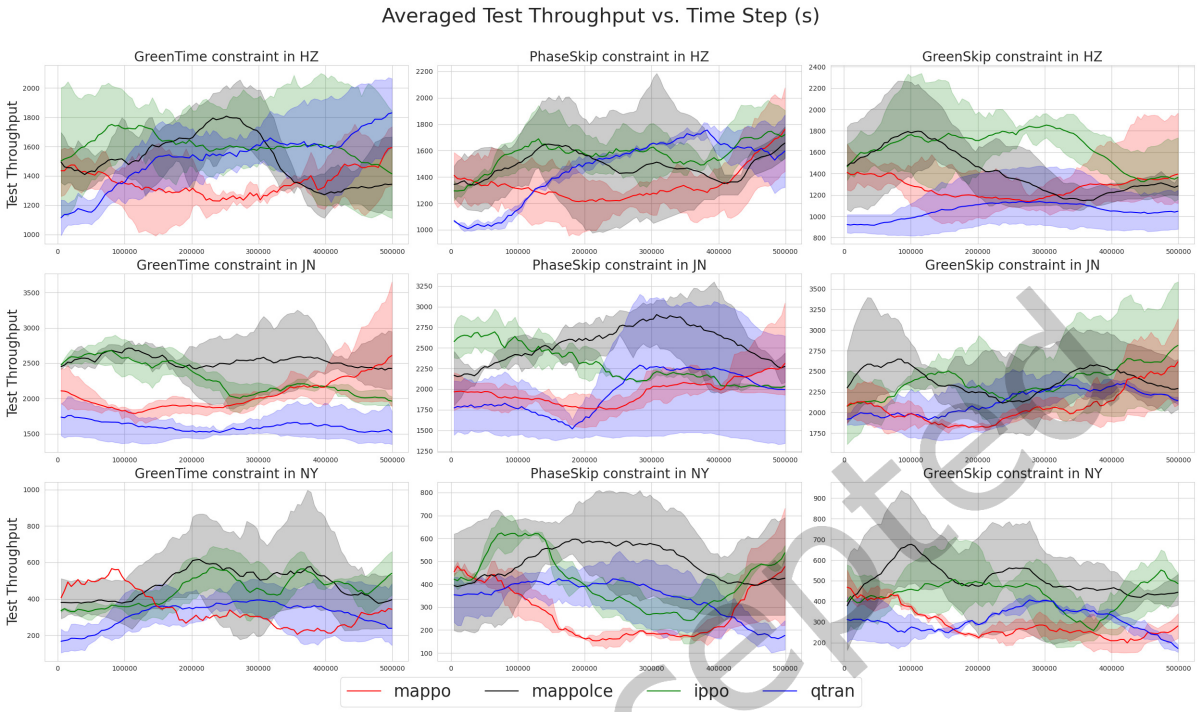

Fig. 11. Plot of throughput on the test set over 500,000 timesteps for the MAPPO-LCE algorithm compared to baseline algorithms across all environments and constraints. Plots are best viewed in color. Plots are best viewed in color.

Environment	MAPPO-LCE % Test Reward increase over comparison algorithms		
	MAPPO	IPPO	QTRAN
HZ	**10.31%**	0.63%	6.76%
JN	**3.5%**	1.0%	0.1%
NY	5.48%	-0.96%	**5.53%**

Table 5. Comparison of the Test Reward between MAPPO-LCE and MARL baseline algorithms using a sum of all constraints for each traffic environment.

on the communication graph, such as [12, 14, 19, 28] would help avoid redundant communication that would perversely affect agent learning.

Another idea is to expand our constraints. Formulating our current constraints as hard constraints and adding additional soft constraints such as variance in throughput or waiting times could more closely represent real-world environments while creating a model that values fairness and safety. In addition, we could add further constraints by expanding the environment to include different types of vehicles, such as buses, ambulances, and trams, to develop more generalizable traffic management strategies that accommodate diverse transportation needs. Adding more significant constraints on their delay and waiting time could lead to more robust constraints that better reflect real-life scenarios.

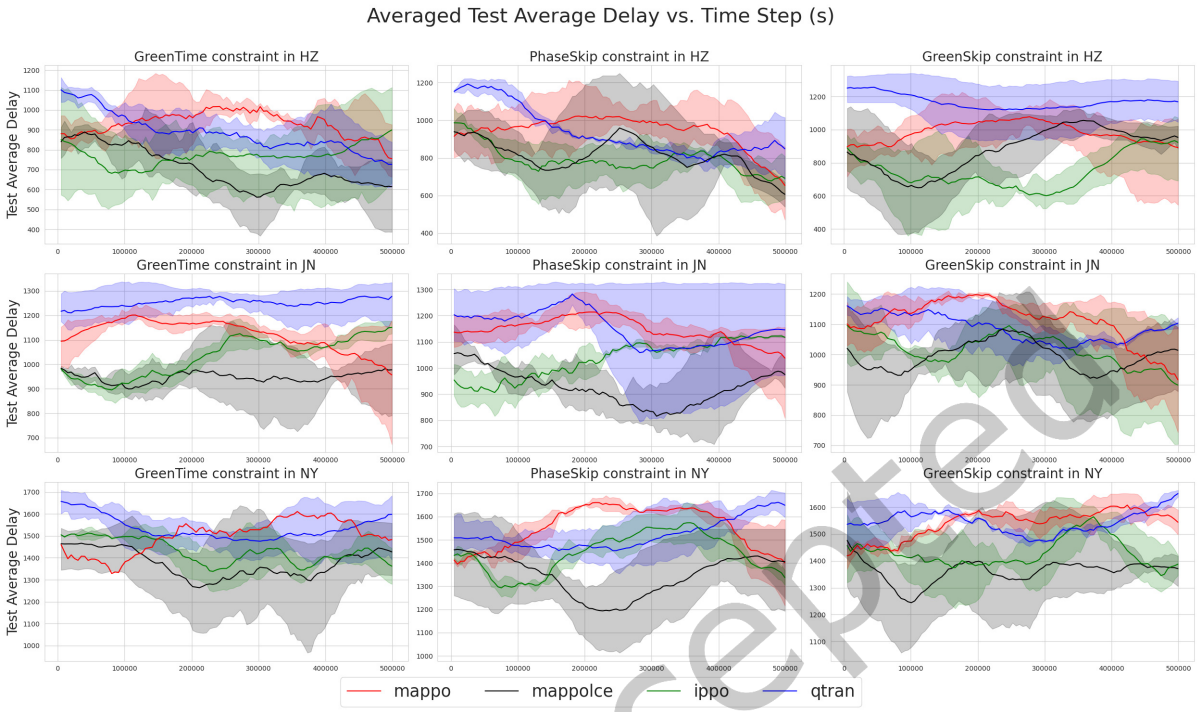

Fig. 12. Plot of average delay on the test set over 500,000 timesteps for the MAPPO-LCE algorithm compared to baseline algorithms across all environments and constraints. Plots are best viewed in color.

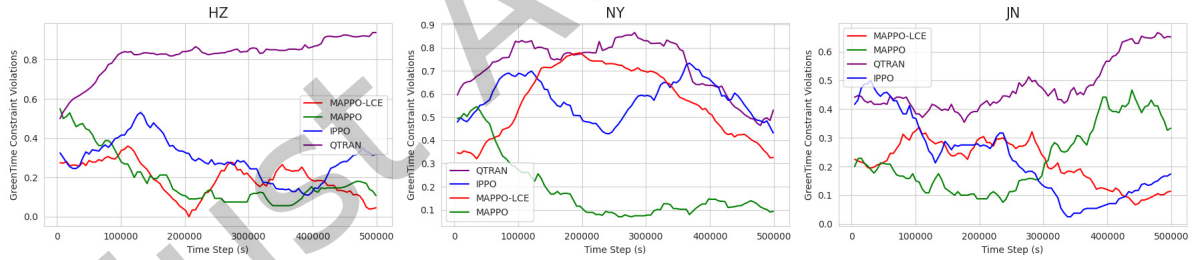

Fig. 13. Plot of test Green Time constraint values over all environments and algorithms. Plots are best viewed in color.

Expanding the ways our algorithm treats constraints and is able to incorporate them is also another route for future expansion. For example, one aspect of our algorithm is that the constraint value is globalized and applied to all agents equally, which means we only need one cost critic and cost estimator. However, this means we cannot model each agent to correct any individual constraint violations. Future work serves to accurately incorporate such fine-grained control over constraint violations without the significant overhead of training cost estimators for each agent.

While ATSC is a partially observable Markov Game, a final idea is to give each agent a better idea of their surroundings through expectation alignment. ELIGN [25] is a method for multi-agent expectation alignment that

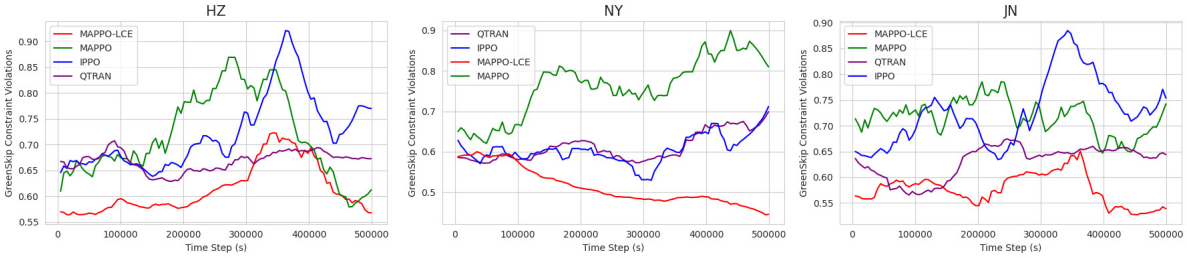

Fig. 14. Plot of test Green Skip constraint values over all environments and algorithms. Plots are best viewed in color.

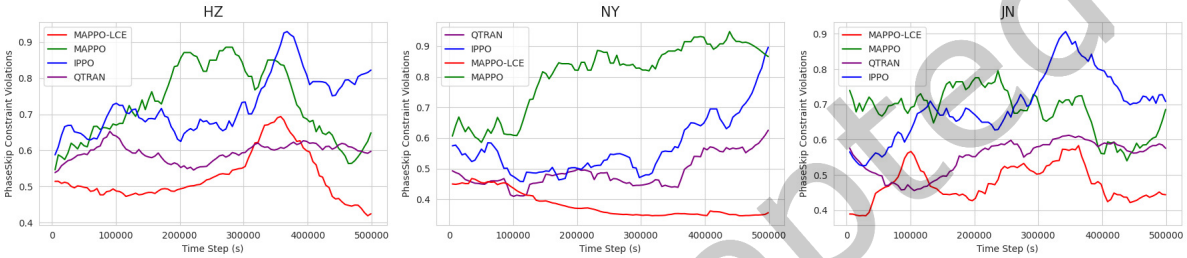

Fig. 15. Plot of test Phase Skip constraint values over all environments and algorithms. Plots are best viewed in color.

Environment	MAPPO-LCE % Throughput increase over comparison algorithms		
	MAPPO	IPPO	QTRAN
HZ	**17.75%**	-0.73%	0.37%
JN	**7.25%**	2.66%	1.39%
NY	**36.93%**	-8.69%	24.84%

Table 6. Comparison of the Test Throughput metric between MAPPO-LCE and MARL baseline algorithms using a sum of all constraints for each traffic environment.

Environment	MAPPO-LCE % Average Delay decrease over comparison algorithms		
	MAPPO	IPPO	QTRAN
HZ	**14.87%**	-0.99%	6.12%
JN	**4.74%**	1.41%	1.44%
NY	5.19%	-2.17%	**5.23%**

Table 7. Comparison of the Test Average Delay metric between MAPPO-LCE and MARL baseline algorithms using a sum of all constraints on each traffic environment.

aligns the shared expectations of an agent to its actual actions through an intrinsic reward. Predicting neighboring agents' actions in a second-order theory of mind approach allows for better coordination and can easily be added

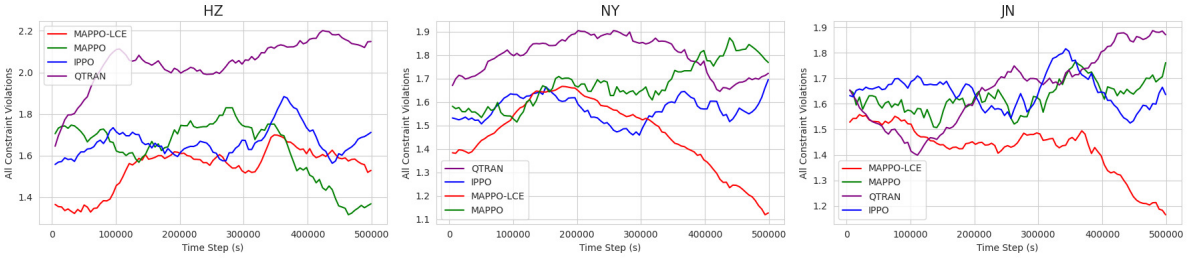

Fig. 16. Plot of summed test constraint values over all environments and algorithms. Plots are best viewed in color.

to existing methods to find more optimal policies. In the ATSC problem, this may allow each agent to predict swells or dips in traffic before they reach the intersection that the agent controls, further increasing its ability to make realistic traffic policies.

8 Conclusion

In this paper, we focus on finding scalable algorithms for the Adaptive Traffic Signal Control problem in real-world traffic environments. We propose a novel algorithm, MAPPO-LCE, for constrained multi-agent reinforcement learning. We expand upon Multi-Agent Proximal Policy Optimization (MAPPO) by incorporating elements of MAPPO-Lagrangian [17] and introducing a Lagrange Cost Estimator to accurately predict constraints even under unstable conditions. While we only focused on three constraints, MAPPO-LCE can be used with any number of general traffic constraints and can be extended to any constrained MARL problem. Our experimental results using the CityFlow environment in multiple real-world settings show that MAPPO-LCE outperforms other baseline methods with suitable constraints. Our findings indicate that constrained multi-agent reinforcement learning can identify more optimal traffic policies for ATSC in real-world conditions and holds strong potential for real-world deployment.

References

[1] Ghadeer Abdelkader, Khalid Elgazzar, and Alaa Khamis. 2021. Connected Vehicles: Technology Review, State of the Art, Challenges and Opportunities. *Sensors* 21, 22 (2021). doi:10.3390/s21227712

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*. PMLR, 22–31.

[3] Fahmy Adan, Yuxiang Feng, Panagiotis Angeloudis, Mohammed Quddus, and Washington Ochieng. 2023. Constrained Multi-Agent Reinforcement Learning Policies for Cooperative Intersection Navigation and Traffic Compliance. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 4079–4085. doi:10.1109/ITSC57777.2023.10422440

[4] Mohammad Aslani, Mohammad Saadi Mesgari, and Marco Wiering. 2017. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C: Emerging Technologies* 85 (2017), 732–752. doi:10.1016/j.trc.2017.09.020

[5] Tianshu Bao, Hua Wei, Junyi Ji, Daniel Work, and Taylor Thomas Johnson. 2024. Spatial-Temporal PDE Networks for Traffic Flow Forecasting. In *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2024, Vilnius, Lithuania, September 9–13, 2024, Proceedings, Part X* (Vilnius, Lithuania). Springer-Verlag, Berlin, Heidelberg, 166–182. doi:10.1007/978-3-031-70381-2_11

[6] Dimitri P. Bertsekas. 1996. *Constrained Optimization and Lagrange multiplier methods.* Athena Scientific.

[7] G Brockman. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* (2016).

[8] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. 2020. Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (Apr. 2020), 3414–3421. doi:10.1609/aaai.v34i04.5744

[9] Yue Chen, Changle Li, Wenwei Yue, Hehe Zhang, and Guoqiang Mao. 2021. Engineering A Large-Scale Traffic Signal Control: A Multi-Agent Reinforcement Learning Approach. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 1–6. doi:10.1109/INFOCOMWKSHPS51825.2021.9484451

[10] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. 2007. *Self-Organizing Traffic Lights: A Realistic Simulation.* Springer London, 41–50. doi:10.1007/978-1-84628-982-8_3

[11] Eddie Curtis. 2017. EDC-1: Adaptive Signal Control Technology | Federal Highway Administration. https://www.fhwa.dot.gov/innovation/everydaycounts/edc-1/asct.cfm

[12] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. 2020. TarMAC: Targeted Multi-Agent Communication. arXiv:1810.11187 [cs.LG] https://arxiv.org/abs/1810.11187

[13] 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).

[14] Shifei Ding, Wei Du, Ling Ding, Lili Guo, and Jian Zhang. 2024. Learning Efficient and Robust Multi-Agent Communication via Graph Information Bottleneck. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 16 (Mar. 2024), 17346–17353. doi:10.1609/aaai.v38i16.29682

[15] Mohamed Essa and Tarek Sayed. 2020. Self-learning adaptive traffic signal control for real-time safety optimization. *Accident Analysis & Prevention* 146 (2020), 105713. doi:10.1016/j.aap.2020.105713

[16] Hankang Gu, Shangbo Wang, Xiaoguang Ma, Dongyao Jia, Guoqiang Mao, Eng Gee Lim, and Cheuk Pong Ryan Wong. 2024. Large-Scale Traffic Signal Control Using Constrained Network Partition and Adaptive Deep Reinforcement Learning. *Trans. Intell. Transport. Sys.* 25, 7 (April 2024), 7619–7632. doi:10.1109/TITS.2024.3352446

[17] Shangding Gu, Jakub Grudzien Kuba, Munning Wen, Ruiqing Chen, Ziyan Wang, Zheng Tian, Jun Wang, Alois Knoll, and Yaodong Yang. 2022. Multi-Agent Constrained Policy Optimisation. arXiv:2110.02793 [cs.AI] https://arxiv.org/abs/2110.02793

[18] Ammar Haydari, Vaneet Aggarwal, Michael Zhang, and Chen-Nee Chuah. 2024. Constrained Reinforcement Learning for Fair and Environmentally Efficient Traffic Signal Controllers. *ACM J. Auton. Transport. Syst.* 2, 1, Article 2 (Sept. 2024), 19 pages. doi:10.1145/3676169

[19] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. 2024. Learning Multi-Agent Communication from Graph Modeling Perspective. arXiv:2405.08550 [cs.LG] https://arxiv.org/abs/2405.08550

[20] Xingshuai Huang, Di Wu, and Benoit Boulet. 2023. Fairness-Aware Model-Based Multi-Agent Reinforcement Learning for Traffic Signal Control. https://openreview.net/forum?id=sy0PqUr2fq9

[21] Jiechuan Jiang and Zongqing Lu. 2019. Learning fairness in multi-agent systems. *Advances in Neural Information Processing Systems* 32 (2019).

[22] 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 2022, Vienna, Austria, 23-29 July 2022*, Luc De Raedt (Ed.). ijcai.org, 3854–3860. doi:10.24963/ijcai.2022/535

[23] Michael L. Littman. 2001. Friend-or-Foe Q-learning in General-Sum Games. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 322–328.

[24] Songtao Lu, Kaiqing Zhang, Tianyi Chen, Tamer Başar, and Lior Horesh. 2021. Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 8767–8775.

[25] Zixian Ma, Rose Wang, Fei-Fei Li, Michael Bernstein, and Ranjay Krishna. 2022. Elign: Expectation alignment as a multi-agent intrinsic reward. *Advances in Neural Information Processing Systems* 35 (2022), 8304–8317.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602 [cs.LG] https://arxiv.org/abs/1312.5602

[27] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. 2017. Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning. arXiv:1704.08883 [cs.LG] https://arxiv.org/abs/1704.08883

[28] Yaru Niu, Rohan Paleja, and Matthew Gombolay. 2021. Multi-Agent Graph-Attention Communication and Teaming. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 964–973.

[29] Afshin Oroojlooy, Mohammadreza Nazari, Davood Hajinezhad, and Jorge Silva. 2020. AttendLight: Universal Attention-Based Reinforcement Learning Model for Traffic Signal Control. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 4079–4090. https://proceedings.neurips.cc/paper_files/paper/2020/file/29e48b79ae6fc68e9b6480b677453586-Paper.pdf

[30] Hali Pang and Weilong Gao. 2019. Deep Deterministic Policy Gradient for Traffic Signal Control of Single Intersection. In *2019 Chinese Control And Decision Conference (CCDC)*. 5861–5866. doi:10.1109/CCDC.2019.8832406

[31] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*. http://arxiv.org/abs/2006.07869

[32] Majid Raeis and Alberto Leon-Garcia. 2021. A Deep Reinforcement Learning Approach for Fair Traffic Signal Control. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)* (Indianapolis, IN, USA). IEEE Press, 2512–2518. doi:10.1109/ITSC48978.2021.9564847

[33] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv:1803.11485 [cs.LG] https://arxiv.org/abs/1803.11485

[34] Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. 2019. Reinforcement Learning with Explainability for Traffic Signal Control. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 3567–3572. doi:10.1109/ITSC.2019.8917519

[35] Umer Siddique, Paul Weng, and Matthieu Zimmer. 2020. Learning Fair Policies in Multi-Objective (Deep) Reinforcement Learning with Average and Discounted Rewards. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 8905–8915. https://proceedings.mlr.press/v119/siddique20a.html

[36] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 5887–5896.

[37] Daniel Tabas, Ahmed S Zamzam, and Baosen Zhang. 2023. Interpreting Primal-Dual Algorithms for Constrained Multiagent Reinforcement Learning. In *Learning for Dynamics and Control Conference*. PMLR, 1205–1217.

[38] Shijie Wang and Shangbo Wang. 2023. A Novel Multi-Agent Deep RL Approach for Traffic Signal Control. arXiv:2306.02684 [cs.AI] https://arxiv.org/abs/2306.02684

[39] Tianyu Wang, Teng Liang, Jun Li, Weibin Zhang, Yiji Zhang, and Yan Lin. 2020. Adaptive Traffic Signal Control Using Distributed MARL and Federated Learning. In *2020 IEEE 20th International Conference on Communication Technology (ICCT)*. 1242–1248. doi:10.1109/ICCT50939.2020.9295660

[40] Ziyan Wang, Meng Fang, Tristan Tomilin, Fei Fang, and Yali Du. 2024. Safe Multi-agent Reinforcement Learning with Natural Language Constraints. *arXiv preprint arXiv:2405.20018* (2024).

[41] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. 2019. PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1290–1298. doi:10.1145/3292500.3330949

[42] 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* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1913–1922. doi:10.1145/3357384.3357902

[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] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*. 3620–3624.

[45] Yuli Zhang, Shangbo Wang, Xiaoguang Ma, Wenwei Yue, and Ruiyuan Jiang. 2023. Large-Scale Traffic Signal Control by a Nash Deep Q-network Approach. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 4584–4591. doi:10.1109/ITSC57777.2023.10422534

[46] 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* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1963–1972. doi:10.1145/3357384.3357900

[47] Guanjie Zheng, Xinshi Zang, Nan Xu, Hua Wei, Zhengyao Yu, Vikash Gayah, Kai Xu, and Zhenhui Li. 2019. Diagnosing Reinforcement Learning for Traffic Signal Control. arXiv:1905.04716 [cs.LG] https://arxiv.org/abs/1905.04716

[48] Wei Zhou, Dong Chen, Jun Yan, Zhaojian Li, Huilin Yin, and Wanchen Ge. 2022. Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. *Autonomous Intelligent Systems* 2, 1 (March 2022). doi:10.1007/s43684-022-00023-5

A Hyperparameter Selection

A.1 Algorithm Hyperparameters

We use the same hyperparameters since all of our baseline algorithms are adapted from the ePYMARL library [31]. We provide a full list of algorithmic environment hyperparameters in Table 8. For the constraint trade-off hyperparameter ζ, we set it to 0.2 for all constraints. We model the cost estimator as a Multi-Layer Perceptron

(MLP), with two hidden layers and a hidden layer size of 128. We also use an Adam optimizer with a learning rate of 10^{-4} to train the cost estimator. We set the cost limit for all experiments to 0.

A.2 Environment Hyperparameters

Each time step in the environment is composed of T_g inner steps to update the environment, which we set to 10. Thus, simulating for 500,000 steps is the same as simulating approximately 1400 episodes. After the policy selects an action, each inner step simulates 1 second of the environment. To simulate yellow lights without actually implementing them directly, each traffic light instead turns off all lights that would be switched between phases for T_y time before fully turning them red or green. We set T_y to 5 time steps, which is equivalent to 5 seconds. Each constraint also has a hyperparameter that controls its severity. For constraint thresholds, we set $G_{max\ time}$ to 40, $P_{max\ skips}$ to 16, and $G_{max\ skips}$ to 4.

Table 8. Hyperparameter Comparison of RL Algorithms

Hyperparameter	**IPPO**	**MAPPO**	**QTRAN**	**MAPPO-LCE**
Epsilon Start	1.0	1.0	1.0	1.0
Epsilon Finish	0.05	0.05	0.05	0.05
Epsilon Anneal Time	500000	500000	500000	500000
Learning Rate (lr)	0.00005	0.0005	0.0005	0.00005
Gamma	0.985	0.985	0.985	0.985
Hidden Dim	128	128	32	128
Grad Norm Clip	10	10	5	10
Critic Coef	0.5	0.5	-	0.5
Entropy Coef	0	0	-	0
Reg Coef	0.01	0.01	-	0.01
GAE Lambda	0.95	0.95	-	0.95
Mini Epochs	2	2	1	2
Eps Clip	0.15	0.15	-	0.15
Target Update Interval	200	200	200	200
Mixing Embed Dim	-	-	64	-
Opt Loss	-	-	1	-
Nopt Loss	-	-	0.1	-
Lambda Init	-	-	-	0.01
Lambda LR	-	-	-	0.0001
Batch Size	8	8	8	8
Buffer Size	8	8	8	8

Received 29 March 2025; revised 3 August 2025; accepted 7 September 2025