Feudal Multi-Agent Deep Reinforcement Learning for Traffic Signal Control[*]

Jinming Ma
School of Computer Science and Technology,
University of Science and Technology of China
Hefei, Anhui, China
jinmingm@mail.ustc.edu.cn

Feng Wu[†]
School of Computer Science and Technology,
University of Science and Technology of China
Hefei, Anhui, China
wufeng02@ustc.edu.cn

ABSTRACT

Reinforcement learning (RL) is a promising technique for optimizing traffic signal controllers that dynamically respond to real-time traffic conditions. Recent efforts that applied Multi-Agent RL (MARL) to this problem have shown remarkable improvement over centralized RL, with the scalability to solve large problems by distributing the global control to local RL agents. Unfortunately, it is also easy to get stuck in local optima because each agent only has partial observability of the environment with limited communication. To tackle this, we borrow ideas from feudal RL and propose a novel MARL approach combining with the feudal hierarchy. Specifically, we split the traffic network into several regions, where each region is controlled by a manager agent and the agents who control the traffic signals are its workers. In our method, managers coordinate their high-level behaviors and set goals for their workers in the region, while each lower-level worker controls traffic signals to fulfill the managerial goals. By doing so, we are able to coordinate globally while retain scalability. We empirically evaluate our method both in a synthetic traffic grid and real-world traffic network using the SUMO simulator. Our experimental results show that our approach outperforms the state-of-the-art in almost all evaluation metrics commonly used for traffic signal control.

CCS CONCEPTS

• **Computing methodologies → Cooperation and coordination**; **Multi-agent reinforcement learning**; *Multi-agent systems*; *Dynamic programming for Markov decision processes*;

KEYWORDS

Multi-Agent Reinforcement Learning; Deep Reinforcement Learning; Feudal Reinforcement Learning; Traffic Signal Control;

ACM Reference Format:
Jinming Ma and Feng Wu. 2020. Feudal Multi-Agent Deep Reinforcement Learning for Traffic Signal Control. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

[*]This work was supported in part by the National Key R&D Program of China (Grant No. 2017YFB1002204), the National Natural Science Foundation of China (Grant No. U1613216, Grant No. 61603368), and the Guangdong Province Science and Technology Plan (Grant No. 2017B010110011).
[†]Feng Wu is the corresponding author.

1 INTRODUCTION

Traffic congestion in metropolitan areas has been becoming a worldwide problem as a consequence of population growth and urbanization. *Reinforcement Learning* (RL) and its deep learning counterpart have shown promising results [3–5, 8, 12, 15, 16, 18, 23, 25] on reducing potential congestions in traffic networks, by learning policies for traffic light controllers that dynamically respond to real-time traffic conditions. Unlike traditional model-driven approaches [9, 19], RL does not rely on heuristic assumptions or expert knowledge. Instead, it formulates the traffic signal control problem as a *Markov Decision Process* (MDP) and learns the optimal policy based on the experience interacting with complex traffic systems.

However, a centralized RL approach is usually infeasible for large traffic signal control problems because: 1) Collecting all traffic measurements in the network to form a global state will cause high latency in practice; 2) The joint action space of the agents grows exponentially in the number of signalized intersections. Therefore, it is more efficient and natural to formulate traffic signal control as a cooperative *Multi-Agent RL* (MARL), where each intersection is controlled by a single agent with local observations.

To date, existing work on the multi-agent perspective for traffic signal control either falls back to independent learning [3, 5, 14, 26] or depends on centralized optimization of coordinated agents [6, 11, 30]. Centralized optimization has the scalability issue as it requires maximization over a huge joint action space. Independent RL such as *Independent Q-Learning* (IQL) [26] is scalable, in which each agent only learns its own policy independently by modeling other agents as parts of the environment dynamics. However, the environment becomes non-stationary when other agents update their policies. Furthermore, it is difficult to achieve the global optima when agent only optimizes its own reward based on local observations.

To alleviate this, *Multi-agent Advantage Actor-Critic* (MA2C) [8] was proposed to solve traffic signal control problems. Similar to IQL, MA2C is scalable as each agent only learns its own policy independently. Most importantly, it is more stable and robust than IQL as: 1) It includes the observations and fingerprints of neighboring agents in the state so that each agent has more information about cooperative strategy; 2) It introduces a spatial discount factor to scale down the observation and reward signals of neighboring agents so that each agent focuses more on improving nearby traffic. Although the proposed methods stabilize training and guarantee convergence, it is easy to get stuck in local optima due to lack of global coordination. This becomes severe especially when traffic requires to flow across a large area. In such cases, neighborhood adjustment may not be efficient to minimize traffic congestion.

Against this background, we borrow ideas from feudal RL [2, 10, 24] to improve global coordination among agents in the domain of traffic signal control. With feudal hierarchy, a *manager* agent makes decisions in the high level and communicates its goals to multiple lower-level *worker* agents, who are rewarded for achieving the managerial goals. This hierarchical structure is useful especially when local agents only have limited view of the world as in the scenario of traffic signal control. Notice that this is widely used in our society for improving team performance. For instance, in soccer games, coach who has better view of the game frequently communicates her decisions to the team playing in the field. This coach-player structure is effective and even critical to succeed in a competition. We will demonstrate the benefit of such feudal hierarchy for traffic signal control later in the experiments.

Here, we propose *Feudal Multi-agent Advantage Actor-Critic* (FMA2C), which is an extension of MA2C with feudal hierarchy for traffic signal control. To solve the problem, we split the traffic network into several regions, where each region is controlled by a manager agent and the local agents who controls the traffic lights in the region are its workers. Each manager coordinates with other managers and takes an action that corresponds to a goal for its workers. By receiving the goal from its manager, each worker try to achieve the managerial goal while maximizing its own local reward. In our algorithm, managers learn to coordinate in the high level and set goals for their workers, while workers learn to choose actions that fulfill both the managerial goals and its local objectives. To the best of our knowledge, this is the *first* approach to combine MA2C with feudal hierarchy for traffic signal control. In the experiments, we tested our algorithm both in a synthetic traffic grid and real-world traffic network of Monaco city. Our experimental results show that FMA2C outperformed MA2C and other baselines (e.g., IQL, Greedy) in almost all criteria, including queue length, intersection delay, vehicle speed, trip completion flow, and trip delay, commonly used to evaluate the overall traffic conditions.

2 BACKGROUND

This section briefly describes some building blocks of our approach. We build our method based on multi-agent A2C and combine the idea of feudal RL to solve the traffic signal control problems.

2.1 Multi-Agent A2C

Policy gradient is a RL method that directly optimizes the parameterized policy π_θ with experience trajectories. Early work (e.g., REINFORCE) uses the estimated return $\hat{R}_t = \sum_{\tau=t}^{T-1} \gamma^{\tau-t} r_\tau$ to minimize the policy loss function:

$$\mathcal{L}(\theta) = -\frac{1}{|B|} \sum_{t \in B} \log \pi_\theta(a_t|s_t)\hat{R}_t \tag{1}$$

where $B = \{(s_t, a_t, r_t)\}$ is the experience replay buffer. However, it suffers from high variance as $\hat{R}_t$ is very noisy.

Advantage Actor-Critic (A2C) improves the policy gradient by introducing the advantage value $A_t = \tilde{R}_t - V_{\omega^-}(s_t)$ where $\tilde{R}_t = \hat{R}_t + \gamma^{T-t} V_{\omega^-}(s_T)$. It reduces the bias of sampled return by adding the value of the last state. Given this, the actor minimizes the policy loss function to update the policy parameter θ as below:

$$\mathcal{L}(\theta) = -\frac{1}{|B|} \sum_{t \in B} \log \pi_\theta(a_t|s_t)A_t \tag{2}$$

In turn, the critic minimizes the value loss function to update the value parameter ω as follow:

$$\mathcal{L}(\omega) = \frac{1}{2|B|} \sum_{t \in B} (\tilde{R}_t - V_\omega(s_t))^2 \tag{3}$$

To extend single-agent A2C to multi-agent settings, a straightforward idea is for each agent to independently learn its own policy π_{θ_i} and the corresponding value function V_{ω_i}. If the global reward and state are shared among agents, the local return of agent i can be estimated with the other agents' policies $\pi_{\theta_{-i}}$ fixed as:

$$\tilde{R}_{t,i} = \hat{R}_t + \gamma^{T-t} V_{\omega_i^-}(s_T|\pi_{\theta_{-i}^-}) \tag{4}$$

When global information sharing is infeasible due to communication latency, one idea is to consider the communication only between neighboring agents in order to coordinate the agents' behavior. Specifically, the local state of each agent i is augmented with the local states of its neighbors: $s_{t,\mathcal{U}_i} = \{s_{t,j}\}_{j \in \mathcal{U}_i}$. Then, each agent i minimizes the value loss with its parameter ω_i as:

$$\mathcal{L}(\omega_i) = \frac{1}{2|B|} \sum_{t \in B} (\tilde{R}_{t,i} - V_{\omega_i}(s_{t,\mathcal{U}_i}))^2 \tag{5}$$

Given the advantage value $A_{t,i} = \tilde{R}_{t,i} - V_{\omega_i^-}(s_{t,\mathcal{U}_i})$, each agent i now minimizes the policy loss to update its parameter θ_i as:

$$\mathcal{L}(\theta_i) = -\frac{1}{|B|} \sum_{t \in B} \log \pi_{\theta_i}(a_{t,i}|s_{t,\mathcal{U}_i})A_{t,i} \tag{6}$$

However, when the other agents' policies π_{-i} is actively updated, the policy gradient may be inconsistent across mini-batches since the advantage is conditioned on changing policy parameters θ_{-i}.

To stabilize the training, *Multi-agent A2C* (MA2C) [8] proposes two techniques: 1) it includes information of neighborhood policies to improve the observability of each local agent; 2) it uses a spatial discount factor to weaken the state and reward signals from other agents. However, it is easy to get stuck in local optima because each agent still has very limited range of sight. To address this, we borrow ideas from feudal RL and make an improvement.

2.2 Feudal RL

Feudal RL [10] speeds up RL by enabling simultaneous learning at multiple resolutions in space and time. It creates a control hierarchy where high-level managers learn how to set goals for their workers who, in turn, learn how to satisfy them. Here, a goal is simply an action of the managers to communicate with workers and define their reward functions. With this settings, the manager must learn to communicate its goals judiciously in order to solve global problem. In turn, the workers must learn how to act in the light of the resulting managerial reward, in addition to immediate rewards they might also experience from the environment. This framework has been extended to use neural networks [24] and multi-agent settings [2]. In this paper, we use a feudal hierarchy similar to feudal multi-agent hierarchy proposed by [2]. However, we develop a different mechanism for managers to communicate their goals to their workers, which is suitable for traffic signal control.

3 RELATED WORK

RL has been extensively studied in traffic signal control. Early work mostly apply RL to single traffic light control [1, 7, 20, 22, 27]. The key challenge focused by them is how to represent the high-dimensional continuous states in the Q-function under complex traffic dynamics. Recently, deep RL approaches were implemented to incorporate high dimensional state information into a more realistic settings in traffic signal control [4, 5, 12, 15, 16, 18, 25].

Existing work on multi-agent traffic signal control mostly relied on *Independent Q-Learning* (IQL). Wiering [26] applied model-based IQL to each intersection. Chu et al. [5] dynamically cluster regions and use IQL to solve the traffic signal control for each region. Aziz et al. [3] improved the observability of IQL by neighborhood information sharing. Van der Pol and Oliehoek [23] applying a Q function learned on a sub-problem to the full problem with transfer planning and max-plus action-selection. Liu et al. [17] proposed distributed multi-agent Q-learning by sharing information about neighboring agents. Alternatively, *coordinated Q-Learning* was also implemented with various message-passing methods [6, 11, 30].

Most recently, Chu et al. [8] proposed MA2C that shows to outperform other methods and is currently the leading approach for traffic signal control. We build our algorithm based on MA2C and make further improvements as describe next.

4 FEUDAL MULTI-AGENT ADVANTAGE ACTOR-CRITIC ALGORITHM

This section introduces our algorithm that extends MA2C to combine with feudal hierarchy. We start with a formal definition of the traffic network with hierarchical structure. Then, we introduce the Markov games that we use to model the control problems of managers and workers. Finally, we propose FMA2C to simultaneously learn policies for both managers and workers.

4.1 Traffic Network with Hierarchical Structure

We consider a traffic network $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, where each vertex $v_i \in \mathcal{V}$ corresponds to an intersection and each edge $e = (v_i, v_j) \in \mathcal{E}$ represents the road between two intersections $v_i, v_j \in \mathcal{V}$. We assume that the traffic signals in each intersection v_i are controlled by agent i that takes road conditions as input and output operational rules for the traffic lights. The neighborhood of agent i is denoted as $\mathcal{N}_i$ and $\mathcal{U}_i = \mathcal{N}_i \cup \{i\}$. The distance between any two agents $d(i, j)$ is measured as the minimum number of edges connecting them. For example, $d(i, j) = 0$ and $d(i, j) = 1$ for any $j \in \mathcal{N}_i$. We spatially partition the network $\mathcal{G}$ into m disjoint sub-networks $\{\mathcal{V}_1, \cdots, \mathcal{V}_m\}$, where $\forall_{\mathcal{V}_i, \mathcal{V}_j}, \mathcal{V}_i \cap \mathcal{V}_j = \emptyset$, $\cup_{k=1}^m \mathcal{V}_k = \mathcal{G}$, and for $\forall i, j \in \mathcal{V}_k$ there exists a path inside $\mathcal{V}_k$ connecting i and j. We call such sub-network $\mathcal{V}_k \subseteq \mathcal{G}$ a *region* in the traffic network.

We assume that each region $\mathcal{V}_k$ is controlled by a *manager* k and each agent $i \in \mathcal{V}_k$ who controls the traffic signals in the region is called its *worker*. Similarly, the neighborhood of manager k is denoted as $\mathcal{N}_k$ and $\mathcal{U}_k = \mathcal{N}_k \cup \{k\}$. In total, there are m managers and n workers in the traffic network. Here, we consider a tree hierarchy where each worker only reports to a single manager. For simplicity, we just consider two-level manager-worker hierarchies as commonly used in the literature [2, 10, 24], though our work can apply to hierarchies with multiple levels by considering super regions of the traffic network.

4.2 Markov Game for Managers and Workers

We model the control problems for managers and workers as partial observable Markov games. For the n workers, the Markov game is defined by a tuple $\mathcal{M}^W = \langle S^W, \{O_i^W\}, \{A_i^W\}, P^W, R^W \rangle$, where: S^W is the state space, O_i^W is the observation space for worker i, A_i^W is the action space for worker i, $P^W : S^W \times A^W \times S^W \to [0, 1]$ is the transition function, and $R^W : S^W \times A^W \to \Re$ is the reward function. In partial observable settings, the state is hidden and each agent only receives a local observation corresponding to the state. Therefore, the policy for each worker i is a mapping from its local observations to its actions $\pi_i^W : O_i^W \to A_i^W$. The objective is to learn a set of policies, one for each worker, that maximize the accumulated rewards $\sum_{t=1}^T \gamma^t r_t^W$ where γ is a discount factor and T is the horizon. Similarly, we define the Markov game for the m managers by a tuple $\mathcal{M}^M = \langle S^M, \{O_k^M\}, \{A_k^M\}, P^M, R^M \rangle$ and the policy π_k^M for each manager k as described above.

Note that $\mathcal{M}^W$ is a model for the underlying traffic signal control problem, where S^W are the states for the global traffic conditions, O_i^W are the observations of local road conditions received by agent i, and A_i^W are a set of control commands for the traffic lights. The dynamics of the traffic network is modeled by the transition function P^W and the objective of traffic optimization is specified by the reward function R^W. In our experiments, we use the SUMO traffic simulator as our training environment. We will give more details about the model specification in Section 5.

Similar to conventional feudal RL [10], the managers and workers interact in our work with each other through their observations and reward functions. Specifically, each observation of a manager is an abstraction of its workers' observations in the region and each action of a manager corresponds to a goal that needs to be achieved by its workers. The reward of each worker at time step t is augmented by considering the action taken by its manager k as:

$$\hat{r}_{t,i}^W = r_{t,i}^W + \sigma(o_{t,i}^W, a_{t,k}^M) \tag{7}$$

where $r_{t,i}^W$ is the intrinsic reward of worker i and $\sigma : O_i^W \times A_k^M \to \Re$ is a function mapping from the worker's observation and the manager's action to a real number. In practice, there are several ways to define σ. In our experiments, We use the value of the angle between the motion vector of the state and the goal direction vector to measure the degree of following with the goal [24]. Specifically, we use: $C \cdot d_{\cos}(o_{t+1,i}^W - o_{t,i}^W, a_{t,k}^M)$, where $d_{\cos}(X, Y) = X^T Y / (|X||Y|)$ is the cosine similarity between two vectors and C is a constant.

Figure 1 illustrates the framework of manager-worker hierarchy for partially observable Markov games.

4.3 Learning Policies of Managers and Workers

We train managers and workers with FMA2C. The main procedures of our approach are shown in Algorithm 1. At time step t, each manager k samples an action $a_{t,k}^M$ from its current policy $\pi_{t,k}^M$ to produce a goal for the workers in its region $\mathcal{V}_k$. Then, each worker i in $\mathcal{V}_k$ selects an action $a_{t,i}^W$ to execute. After that,

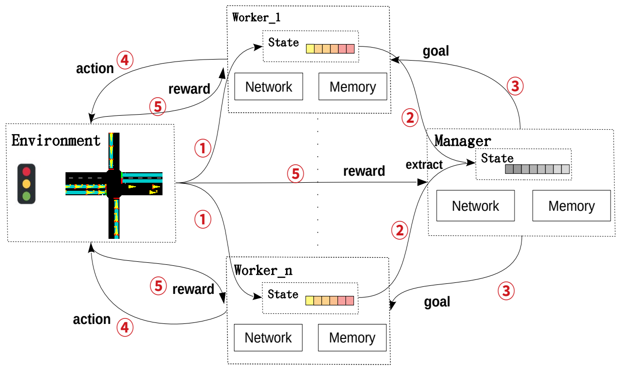


Figure 1: Framework of manager-worker hierarchy.

Algorithm 1: Feudal Multi-Agent Advantage Actor-Critic

```
1  Initialize $\pi^M, \pi^W, B^M \leftarrow \emptyset, B^W \leftarrow \emptyset, t \leftarrow 0, l \leftarrow 0$
2  repeat
3      # Explore experience
4      foreach region $k \in 1..m$ do
5          Sample $a^M_{t,k}$ from $\pi^M_{t,k}$                          ▷ Eq. (8)
6          foreach agent $i \in \mathcal{N}_k$ do
7              Sample $a^W_{t,i}$ from $\pi^W_{t,i}$                      ▷ Eq. (9)
8              Receive $r^W_{t,i}$ and $o^W_{t+1,i}$
9          Receive $r^M_{t,k}$ and $o^M_{t+1,k}$
10     $B^M \leftarrow B^M \cup \{(t, a^M_{t,k}, r^M_{t,k}, o^M_{t+1,k})\}_{k \in 1..m}$
11     $B^W \leftarrow B^W \cup \{(t, a^W_{t,i}, r^W_{t,i}, o^W_{t+1,i})\}_{i \in \mathcal{V}}$
12     $t \leftarrow t+1, l \leftarrow l+1$
13     # Update A2C
14     if $l \geq |B|$ then
15         foreach region $k \in 1..m$ do
16             Estimate $\tilde{R}^M_{t,k}, \forall t \in B^M$                ▷ Eq. (15)
17             Update $\omega^M_k$ with $\eta^M_\omega \nabla \mathcal{L}(\omega^M_k)$        ▷ Eq. (17)
18             Update $\theta^M_k$ with $\eta^M_\theta \nabla \mathcal{L}(\theta^M_k)$        ▷ Eq. (19)
19         foreach agent $i \in \mathcal{V}$ do
20             Estimate $\tilde{R}^W_{t,i}, \forall t \in B^W$                ▷ Eq. (16)
21             Update $\omega^W_i$ with $\eta^W_\omega \nabla \mathcal{L}(\omega^W_i)$        ▷ Eq. (18)
22             Update $\theta^W_i$ with $\eta^W_\theta \nabla \mathcal{L}(\theta^W_i)$        ▷ Eq. (20)
23         $B^W \leftarrow \emptyset, B^M \leftarrow \emptyset, l \leftarrow 0$
24 until stop condition reached.
25 return all managers' and workers' policies $\{\theta^M_k\}$ and $\{\theta^W_i\}$
```

the environment gives corresponding feedback $r^W_{t,i}, o^W_{t+1,i}$ to each worker i. Accordingly, manager k also computes its reward and observation $r^M_{t,k}, o^M_{t+1,k}$. The transitions $(t, a^M_{t,k}, r^M_{t,k}, o^M_{t+1,k})$ and $(t, a^W_{t,i}, r^W_{t,i}, o^W_{t+1,k})$ are stored in replay buffer B^M and B^W respectively. Once the buffer size reaches some predefined size $|B|$, we then use mini-batch gradient descent to update each manager and worker's actor-critic networks. Finally, the training process is terminated when reaching the stop condition.

Inspired by MA2C [8], under limited communication, we augment each manager's observation with the observations of its neighbors $o^M_{t,\mathcal{U}_k} = \{o^M_{t,j}\}_{j \in \mathcal{U}_k}$. Besides, we include a *fingerprint*[1] of its neighbors' latest policies $\pi^M_{t-1,\mathcal{N}_k} = [\pi^M_{t-1,j}]_{j \in \mathcal{N}_k}$. The local policy of manager k with the latest policy parameters θ^-_k is calculated as:

$$\pi^M_{t,k} = \pi_{\theta^-_k}(\cdot | o^M_{t,\mathcal{U}_k}, \pi^M_{t-1,\mathcal{N}_k}) \tag{8}$$

For each worker i, we only consider its neighbors inside the region k, i.e., $\mathcal{N}^-_i = \mathcal{N}_i \cap \mathcal{V}_k$ and $\mathcal{U}^-_i = \mathcal{N}^-_i \cup \{i\}$. Similarly, we augment the worker's observation by its regional neighbors' observations $o^W_{t,\mathcal{U}^-_i} = \{o^W_{t,j}\}_{j \in \mathcal{U}^-_i}$ and latest policy fingerprints $\pi^W_{t-1,\mathcal{N}^-_i} = [\pi^W_{t-1,j}]_{j \in \mathcal{N}^-_i}$. Now, the local policy of each worker i with the latest policy parameters θ^-_i is calculated as:

$$\pi^W_{t,i} = \pi_{\theta^-_i}(\cdot | o^W_{t,\mathcal{U}^-_i}, \pi^W_{t-1,\mathcal{N}^-_i}) \tag{9}$$

Here, we can include the actions taken by its manager and neighboring managers $a^M_{t,\mathcal{U}_k} = \{a^M_{t,j}\}_{j \in \mathcal{U}_k}$. Note that an action taken by managers correspond to a goal that needs to be fulfilled by their workers. Now, the local policy of each worker i is calculated as:

$$\mu^W_{t,i} = \mu_{\theta^-_i}(\cdot | o^W_{t,\mathcal{U}^-_i}, \mu^W_{t-1,\mathcal{N}^-_i}, a^M_{t,\mathcal{U}_k}) \tag{10}$$

We extend each manager k's reward to consider the discounted rewards of its neighbors $\mathcal{N}_k$, where $\alpha \in [0,1]$ is a discount factor:

$$\tilde{r}^M_{t,k} = r^M_{t,k} + \sum_{j \in \mathcal{N}_k} \alpha \cdot r^M_{t,j} \tag{11}$$

For each worker i, we adjust its reward by including the rewards of its regional members in $\mathcal{V}_k$ and the reward of its manager $\hat{r}^W_{t,j}$. We spatially scale down the reginal members' rewards by a discount factor $\alpha \in [0,1]$. Here, $d(i,j)$ represents the distance between two agents i, j, e.g., $d(i,i) = 0$ and $d(i,j) = 1$ if agent i and j are adjacent. Now, the reward of worker i is calculated as:

$$\tilde{r}^W_{t,i} = \sum_{d=0}^{D_i} \left(\sum_{j \in \mathcal{V}_k | d(i,j) = d} \alpha^d \cdot \hat{r}^W_{t,j} \right) + \tilde{r}^M_{t,k} \tag{12}$$

where D_i is the maximum distance from agent i. In the experiments, we set $D_i = 1$ so agent only needs to collect rewards of its neighbors.

In order to maintain consistency, we also discount the neighborhood observations to produce the information states for managers and workers respectively as follow:

$$\tilde{s}^M_{t,\mathcal{U}_k} = [o^M_{t,k}] \cup \alpha [o^M_{t,j}]_{j \in \mathcal{N}_k} \tag{13}$$

$$\tilde{s}^W_{t,\mathcal{U}_i} = [o^W_{t,i}] \cup \alpha [o^W_{t,j}]_{j \in \mathcal{N}^-_i} \tag{14}$$

With managers' and workers' immediate rewards defined above, the accumulated rewards for them are $\hat{R}^M_{t,k} = \sum_{\tau=t}^{T-1} \gamma^{\tau - t} \tilde{r}^M_{\tau,k}$ and $\hat{R}^W_{t,i} = \sum_{\tau=t}^{T-1} \gamma^{\tau - t} \tilde{r}^W_{\tau,i}$ respectively. Given the approximate state values under the previous parameters, the estimating local returns of managers and workers are as follow:

[1]Similar to MA2C, we use the probability simplex of a policy as its fingerprint.

$$\tilde{R}^M_{t,k} = \hat{R}^M_{t,k} + \gamma^{T-t} V^M_{\omega^-_k}(\tilde{s}^M_{T,\mathcal{U}_k}, \pi^M_{T-1,\mathcal{N}_k} | \pi^M_{\theta^-_{-k}}) \quad (15)$$

$$\tilde{R}^W_{t,i} = \hat{R}^W_{t,i} + \gamma^{T-t} V^W_{\omega^-_i}(\tilde{s}^W_{T,\mathcal{U}_i}, \pi^W_{T-1,N_i} | \pi^W_{\theta^-_{-i}}) \quad (16)$$

Now, we can minimize the value loss function for manager k with parameter ω^M_k, with each mini-batch B^M that contains the experience trajectory, as below:

$$\mathcal{L}(\omega^M_k) = \frac{1}{2|B|} \sum_{t \in B^M} \left(\tilde{R}^M_{t,k} - V^M_{\omega_k}(\tilde{s}^M_{t,\mathcal{U}_k}, \pi^M_{t-1,\mathcal{N}_k}) \right)^2 \quad (17)$$

Similarly, we can minimize the value loss function for worker i with parameter ω^W_i with each mini-batch B^W as follow:

$$\mathcal{L}(\omega^W_i) = \frac{1}{2|B|} \sum_{t \in B^W} \left(\tilde{R}^W_{t,i} - V^W_{\omega_i}(\tilde{s}^W_{t,\mathcal{U}_i}, \pi^W_{t-1,N_i}) \right)^2 \quad (18)$$

As aforementioned, we use advantage actor-critic to update the agents' policy parameters. For manager k, the policy loss function with parameter θ^M_k that we try to minimize is as below:

$$\begin{aligned} \mathcal{L}(\theta^M_k) = -\frac{1}{|B|} \sum_{t \in B^M} \Bigg(& \log \pi^M_{\theta_k}(a^M_{t,k} | \tilde{s}^M_{t,\mathcal{U}_k}, \pi^M_{t-1,\mathcal{N}_k}) \tilde{A}^M_{t,k} \\ & - \beta \sum_{a_k \in A^M_k} \pi^M_{\theta_k} \log \pi^M_{\theta_k}(a_k | \tilde{s}^M_{t,\mathcal{U}_k}, \pi^M_{t-1,\mathcal{N}_k}) \Bigg) \end{aligned} \quad (19)$$

where $\tilde{A}^M_{t,k} = \tilde{R}^M_{t,k} - V^M_{\omega^-_k}(\tilde{s}^M_{t,\mathcal{U}_k}, \pi^M_{t-1,\mathcal{N}_k})$ is the advantage value and the additional regularization term is the entropy loss of policy $\pi^M_{\theta_k}$ for encouraging the early-stage exploration.

Similarly, we minimize the policy loss function for worker i with respect to parameter θ^W_i as follow:

$$\begin{aligned} \mathcal{L}(\theta^W_i) = -\frac{1}{|B|} \sum_{t \in B^W} \Bigg(& \log \pi^W_{\theta_i}(a^W_{t,i} | \tilde{s}^W_{t,\mathcal{U}_i}, \pi^W_{t-1,N_i}) \tilde{A}^W_{t,i} \\ & - \beta \sum_{a_i \in A^W_i} \pi^W_{\theta_i} \log \pi^W_{\theta_i}(a_i | \tilde{s}^W_{t,\mathcal{U}_i}, \pi^W_{t-1,N_i}) \Bigg) \end{aligned} \quad (20)$$

where the advantage value is $\tilde{A}^W_{t,i} = \tilde{R}^W_{t,i} - V^W_{\omega^-_i}(\tilde{s}^W_{t,\mathcal{U}_i}, \pi^W_{t-1,N_i})$.

4.4 Discussion

Note that the communication among managers or workers is the same as MA2C. In the learning phase, they share their observations, rewards, current policy's fingerprints with their neighbors to coordinate their policy updating. During execution, they need to share their observations with the neighbors for selecting an action because the policies are fixed so as the fingerprints. The extra communication introduced by our algorithm is the communication between managers and their workers. In the learning phase, each worker must share their observations with their manager so it can do the abstract and make its own observation. Each manager sends its action (goal) and reward to its workers for them to update their policies. During execution, each worker can select its action with or without message from the manager.

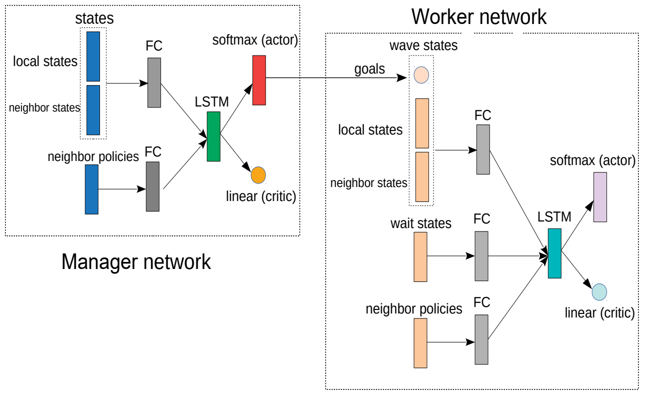


Figure 2: Neural networks for managers and workers.

It is worth pointing out that our algorithm is a decentralized algorithm (similar to MA2C) where each worker computes its policy locally with the information of its neighbors in the region and each manager computes a policy with respect to its workers in the region. Generally, it can scale up to large problems with many levels of hierarchy if necessary. Note that each worker agent takes actions with the local observation on the number of approaching vehicles for all incoming lanes. If unexpected incidents such as accidents and lane closure happen, the agent will act accordingly when the number of approaching vehicles for the corresponding lanes drops.

5 EXPERIMENTS

We implemented our FMA2C algorithm using the SUMO simulator [13] — a commonly used microscopic traffic simulator in the literature. We tested FMA2C in several synthetic traffic grids and a real-world traffic network from Monaco city. We compare our results with MA2C [8], which is currently the leading multi-agent RL approach for traffic signal control and has shown better performance than other baselines. For fair comparison, we ran MA2C with the source code[2] released by the authors and adopt identical parameters. For completeness, we also show results for IQL-LR (IQL with linear regression), IQL-DNN (IQL with deep neural network), and Greedy (agents choose greedy actions) as in the MA2C paper.

5.1 Model Setting

As aforementioned, we model the control problems of managers and workers as partially observable Markov games. There are many possible ways to specify the observation, action and reward function in the literature. We follow the ones used in the MA2C paper [8] for the workers and accordingly create the model for the managers. Next, we describe our model definitions.

5.1.1 Observation Definition. For each worker i, we define the local observation as: $o^W_{t,i} = \{\mathtt{wave}_t[l], \mathtt{wait}_t[l]\}_{l \in L_i}$, where l is each incoming lane of intersection i, `wait` measures the cumulative delay of the first vehicle, and `wave` measures the total number of approaching vehicles along each incoming lane, within 50m to the intersection. For each manager k, we define the local observation:

[2]See: https://github.com/cts198859/deeprl_signal_control

$o^M_{t,k} = \{\texttt{Nwave}_t[l], \texttt{Ewave}_t[l], \texttt{Swave}_t[l], \texttt{Wwave}_t[l]\}_{l \in L_k}$, where l is each lane connect region k to other regions and Nwave, Ewave, Swave, Wwave are the wave for the north, east, south, and west directions respectively.

5.1.2 Action Definition. For each worker, we define the local action as a possible phase (i.e., red-green combinations of traffic lights). Here, we consider five red-green combinations of traffic lights: east-west straight and right-turn phase, east-west left-turn and right-turn phase, and three straight, right-turn and left-turn phases for east, west, and north-south. For each manager, we define the local action as a possible traffic flow. We consider four combinations of north-south and east-west traffic flows.

5.1.3 Reward Definition. For each worker i, we consider both traffic congestion and trip delay and define the local reward as: $r^W_{t,i} = -\sum_{l \in L_i}(\texttt{wave}_{t+\Delta t}[l] + a \cdot \texttt{wait}_{t+\Delta t}[l])$, where a is a tradeoff coefficient. For each manager k, we define the local reward: $r^M_{t,k} = \sum_{l \in L_k} \texttt{arrival}_{t+\Delta t}[l] + \sum_{i \in \mathcal{V}_k} \texttt{liquid}_{t+\Delta t}[i]$, where $\texttt{arrival}_{t+\Delta t}[l]$ is the number of vehicles arriving at the destination within a certain period of time Δt and $\texttt{liquid}_{t+\Delta t}[i]$ is the liquidity of traffic flow within a certain period of time Δt at intersection i.

5.2 Training Setting

Figure 2 illustrates the neural network structures for managers and workers. As shown in the figure, we use similar structure for both managers and workers, where each input is processed by a fully connected (FC) layer. The outputs of the FC layers are integrated and input into a long-short term memory (LSTM). The LSTM layer is useful to extract representations from different state types because the traffic flows are complex spatial-temporal data [8]. The output of the LSTM layer is used as the input for the actor and critic layers. We train all algorithms over 1 million steps, which includes 1400 episodes and each episode's step is 720 (3600 sec in simulation). After training, 10 episodes are simulated to evaluate the policies. For managers, we set $\gamma = 0.96$ and $\alpha = 0.75$, $\eta^M_\theta, \eta^M_w = 2.5e-4$, $|B^M| = 120$, and $\beta = 0.01$. For workers, we set $\gamma = 0.96$,$\alpha = 0.75$, $\eta^M_\theta, \eta^M_w = 2.5e-4$, $|B^M| = 120$, and $\beta = 0.01$. We set $decay = 0.99$ and $\epsilon = 1e-5$ in RMSprop optimizer in both managers and workers.

5.3 Synthetic Traffic Grid

As shown in Figure 3(a), we tested our algorithm in a 4×4 traffic grid formed by two-lane arterial streets with speed limit 20m/s and one-lane avenues with speed limit 11m/s. There are four groups of time-variant flows (i.e., f1, F1, f2, F2) in the simulation and the value of each flow with simulation time is summarized in Table 1. In Figure 3(a), the red lines represent the flow F1 and F2, while the blue lines represent the flow f1 and f2. Specifically, at beginning, flow F1 is generated with origin-destination (O-D) pairs $(x_6 - x_7) \to (x_9 - x_8)$, meanwhile flow f1 is generated with O-D pairs $(x_1 - x_2) \to (x_4 - x_3)$. After 15 minutes, the volumes of F1 and f1 start to decrease, while their opposite flows (with swapped O-D pairs) F2 and f2 start to be generated. We divided the 4×4 traffic grid into 4 regions where each region is controlled by a manager and 4 workers.

Figure 3(d) shows the training curves of each tested algorithm. In the figure, the solid line plots the average reward per training episode: $\bar{R} = \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i \in \mathcal{V}} r^W_{t,i}$ and the shade shows its standard deviation. As we can see from the figure, both FMA2C and MA2C converge to reasonable policies but FMA2C converges faster and more stable than MA2C and the other two baselines.

To test the robustness and generalization of the learned policies, we evaluate them with different traffic flows from the ones that we used to train the policies. Specifically, we made two modification to the test flows: one is to increase the values of the flows (as shown in Table 1) by 50%, the other is to change the (O-D) pairs of the flow to different configuration (as shown in Figure 3(b)). Figures 3(e) and 3(f) show the average queue length and intersection delay respectively for the test case when increasing the flow values. As we can see, FMA2C has smaller queue length and intersection delay than the compared baseline approaches. In Figures 3(g) and 3(h), we observed the same trend with less queue length and intersection delay for FMA2C when changing the flow directions. The overall evaluation metrics for the 4×4 grid with increasing flow values and changing flow directions are summarized in Tables 2(a) and 2(b). As we can see, FMA2C outperforms all the other baselines in almost all evaluation criteria except average trip delay.

Although IQL-LR and Greedy have shorter average trip delay in some problem instances, we observed that IQL-LR and Greedy tend to allow the traffic flow in a lane while keep many vehicles in other lanes to wait for very long time. Specifically, the number of vehicles with waiting time between 1500s and 2000s are 610 and 496 and the number of vehicles with waiting time more than 2000 seconds are 307 and 177 for IQL-LR and Greedy respectively (only 1 vehicle for FMA2C). This is usually undesirable for normal traffic scenarios. As shown in Table 2, it coincidently has the minimum trip delay for some traffic networks (i.e., Tables 2(a-b)) but is not the case for other traffic networks (i.e., Tables 2(c-d)).

We also tested the algorithms in a irregular 4×4 grid by removing some roads and changing traffic flows as shown in Figure 3(c). This can also be viewed as the lanes in the center is closed due to unexpected incidents such as accidents. Specifically, we modified the (O-D) pairs of flows f1 and f2, where f1 and f2 represent $(x_1 - x_2) \to (x_3 - x_4)$ and $(x_3 - x_4) \to (x_1 - x_2)$ respectively. Table 2(c) summarizes the metrics for this scenario. As shown in the table, FMA2C achieves the best performance in all evaluation criteria.

As expected, our algorithm takes additional time for training the managers and computing the fingerprints. Specifically, FMA2C takes 17h43m for training while the learning time for MA2C is 16h22m. The additional training cost (i.e., 1h21m) is relatively small comparing to the total training time. In a traffic simulation with 720 decision cycles, FMA2C totally takes 10.91s for computing the actions while the time for MA2C is 10.34s. The additional execution cost per decision cycle (i.e., 0.57s/720) is neglectable comparing to the time (i.e., 5s) of each decision cycle.

5.4 Monaco Traffic Network

As shown in Figure 3(i), Monaco traffic network, with signalized intersections colored in blue, is a real-world traffic network with 30 intersections extracted from an area of Monaco city. This traffic network has a variety of road and intersection types. The intersections are categorized into 5 types by phase: 11 are two-phase, 4 are three-phase, 10 are four-phase, 1 is five-phase, and 4 are six-phase. In our

Table 1: Time-variant traffic flows within the 4×4 traffic grid.

	0	300	600	900	1200	1500	1800	2100	2400	2700	3000	3300	3600	(sec)
f1	264.0	462.0	594.0	660.0	495.0	330.0	165.0	0	0	0	0	0	0	(veh/h)
F1	440.0	770.0	990.0	1100.0	825.0	550.0	275.0	0	0	0	0	0	0	(veh/h)
f2	0	0	0	166.5	444.0	499.5	555.0	444.0	333.0	111.0	0	0	0	(veh/h)
F2	0	0	0	277.5	740.0	832.5	925.0	740.0	555.0	185.0	0	0	0	(veh/h)

Table 2: Performance for the 4×4 traffic grids and Monaco traffic network with different evaluation metrics.

Metrics	**(a) 4×4 traffic grid (increasing flow value)**					**(b) 4×4 traffic grid (changing flow directions)**				
	FMA2C	MA2C	IQL-DNN	IQL-LR	Greedy	FMA2C	MA2C	IQL-DNN	IQL-LR	Greedy
reward	**-310.22**	-467.65	-850.88	-1647.20	-1940.51	**-302.78**	-406.71	-2007.25	-2420.88	-1867.01
avg. queue length [veh]	**1.72**	2.35	3.31	5.02	5.09	**1.69**	2.23	5.51	6.87	4.78
avg. intersection delay [s/veh]	**14.46**	26.18	87.42	168.10	152.15	**15.62**	25.04	247.32	218.15	154.94
avg. vehicle speed [m/s]	**3.80**	3.27	2.77	2.56	2.80	**3.63**	3.09	1.49	1.18	3.43
trip completion flow [veh/s]	**0.81**	0.79	0.42	0.43	0.50	**0.81**	0.76	0.16	0.16	0.56
trip delay [s]	328	398	359	**273**	296	323	374	450	751	**241**

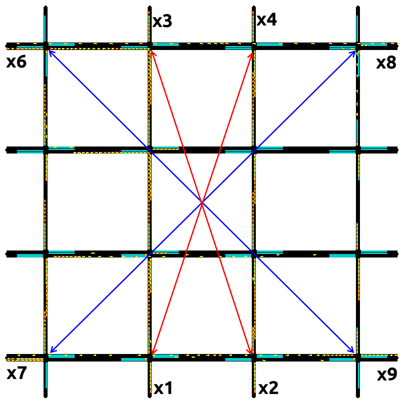
Metrics	**(c) 4×4 traffic grid (irregular grid shape)**					**(d) Monaco traffic network**				
	FMA2C	MA2C	IQL-DNN	IQL-LR	Greedy	FMA2C	MA2C	IQL-DNN	IQL-LR	Greedy
reward	**-105.58**	-138.43	-1527.29	-465.61	-277.27	**-22.77**	-63.21	-100.29	-53.80	-100.29
avg. queue length [veh]	**0.83**	1.21	4.25	2.61	1.08	**0.20**	0.60	1.04	0.54	0.41
avg. intersection delay [s/veh]	**3.86**	4.45	179.90	47.31	19.86	**4.58**	38.07	116.61	97.09	29.90
avg. vehicle speed [m/s]	**4.76**	4.13	2.15	3.79	4.73	**7.53**	4.88	2.38	4.34	7.38
trip completion flow [veh/s]	**0.69**	0.67	0.24	0.57	0.66	**0.68**	0.64	0.54	0.46	0.63
trip delay [s]	**216**	296	268	371	225	**89**	201	267	153	95

experiments, we simulated the traffic flows the same as used in [8]. In Figure 3(i), four traffic flow groups are illustrated by arrows, with origin and destination inside rectangular areas. Specifically, at the first 40min, flows F1 and F2 are simulated following <1, 2, 4, 4, 4, 4, 2, 1> unit flows with 5min intervals, where each unit represents 325veh/h. From 15min to 55min, flows F3 and F4 are simulated. We manually split the 30 intersections in the traffic network into 4 regions with 10, 8, 6, and 6 workers respectively. As an example, we show that different managers can have varying sized assignments of workers in a region.

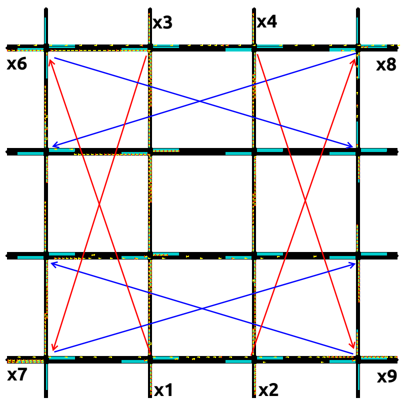
Figure 3(j) shows the training curve of the tested algorithms. As we can see, FMA2C shows a faster and more stable convergence to reasonable policies. Figures 3(k) and 3(l) show the average queue length and average intersection delay of the tested algorithms respectively. As shown in the figure, FMA2C outperforms all the other baselines with less queue length and intersection delay. Table 2(d) summarizes the performance of all the algorithms under different metrics. As we can see, FMA2C has better performance than all the compared algorithms in all evaluation criteria.

6 CONCLUSIONS

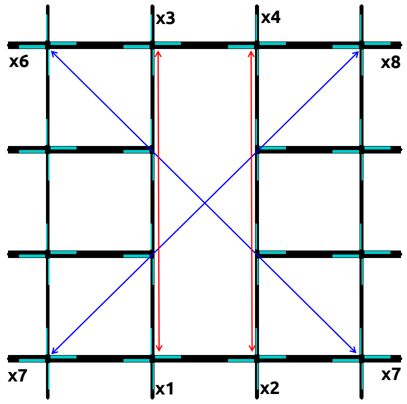
This paper proposed FMA2C, which is an extension of MA2C with feudal hierarchy, to address the global coordination problem in the domain of traffic signal control. To this end, we split the traffic network into regions, where each region is controlled by a manager agent and the agents who control the traffic lights in the region are its workers. Each manager makes decisions in the high level and communicates its goals to the low-level workers, who are responsible to achieve the managerial goals. Our FMA2C algorithm learns policies both for the managers and workers. Experimental results in 4×4 traffic grid and Monaco traffic network using the SUMO simulator show that FMA2C outperforms MA2C and other baselines in almost all evaluate metrics. It is worth noting that our algorithm is a decentralized algorithm where each worker computes its policy locally with the information of its neighbors in the region and each manager computes a policy with respect to its workers in the region. Generally, it can scale up to large problems with many levels of hierarchy if necessary. In the future, we plan to develop algorithms that can optimally split the traffic network into regions and test our algorithm with data from real traffic flow in very large traffic network (e.g., the whole city). Notice that the proposed FMA2C technique is indeed a general decentralized multi-agent reinforcement learning approach, where each agent independently learns its own policy with the guidance of the coordinated regional managers. Apart from the traffic signal control domain, the proposed method has the potential to be applied to other cooperative multi-agent applications, requiring scalable decentralized training, such as multi-robot coordination [28] and disaster management [21, 29].



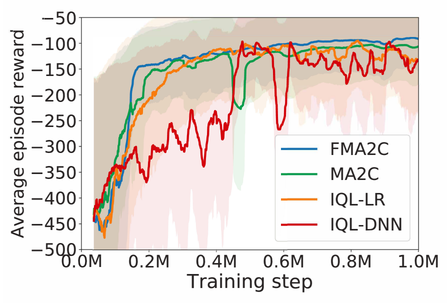
(a) 4×4 synthetic traffic grid



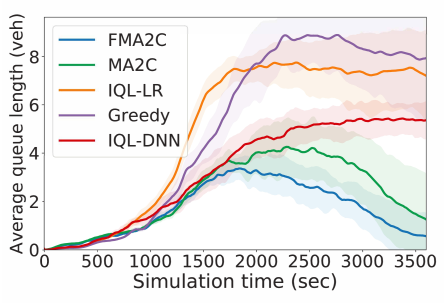
(b) Changing flow direction



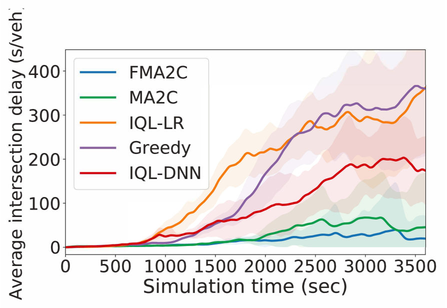
(c) Irregular grid shape



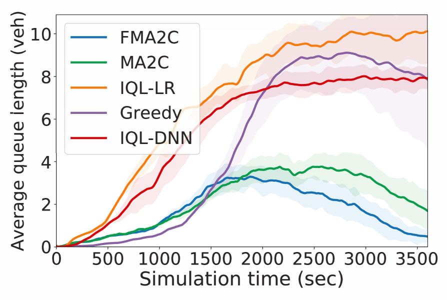
(d) Training curves (4×4 traffic grid)



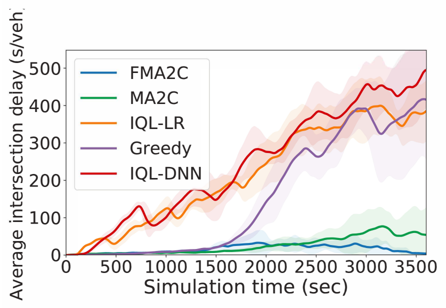
(e) Queue length (increasing flow value)



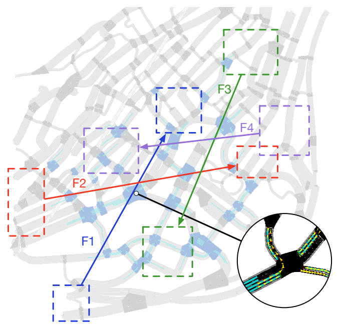
(f) Delay (increasing flow value)



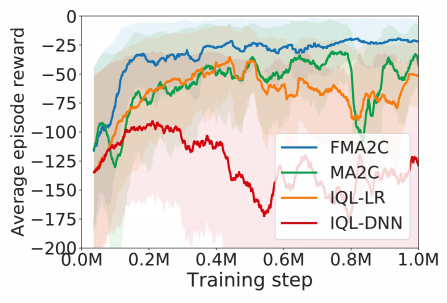
(g) Queue length (changing flow direction)



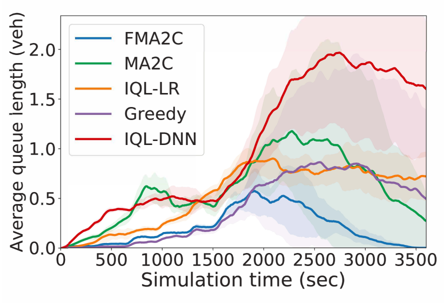
(h) Delay (changing flow direction)



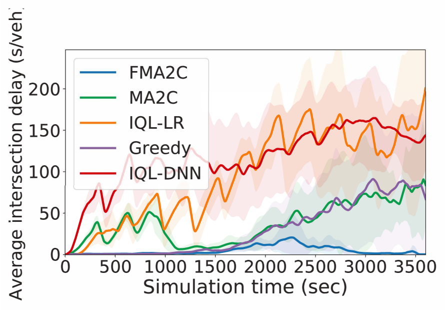
(i) Monaco traffic network



(j) Training curves (Monaco network)



(k) Queue length (Monaco network)



(l) Delay (Monaco network)

Figure 3: Experimental results in the 4×4 synthetic traffic grid and Monaco traffic network. In (a-c), the red lines are the F1, F2 flows and the blue lines are the f1, f2 flows. In (i), four traffic flow groups are shown by arrows, with origin and destination inside rectangular areas. In all line charts, the solid line plots the average value and the shade shows its standard deviation.

REFERENCES

[1] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. 2003. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering* 129, 3 (2003), 278–285.

[2] S Ahilan and P Dayan. 2019. Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning. In *Workshop on Structure & Priors in Reinforcement Learning (SPiRL 2019) at ICLR 2019*. 1–11.

[3] HM Abdul Aziz, Feng Zhu, and Satish V Ukkusuri. 2018. Learning-based traffic signal control algorithms with neighborhood information sharing: An application for sustainable mobility. *Journal of Intelligent Transportation Systems* 22, 1 (2018), 40–52.

[4] Noe Casas. 2017. Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035* (2017).

[5] Tianshu Chu, Shuhui Qu, and Jie Wang. 2016. Large-scale multi-agent reinforcement learning using image-based state representation. In *Proceedings of the 55th IEEE Conference on Decision and Control (CDC)*. 7592–7597.

[6] Tianshu Chu and Jie Wang. 2017. Traffic signal control by distributed Reinforcement Learning with min-sum communication. In *Proceedings of the 2017 American Control Conference (ACC)*. 5095–5100.

[7] Tianshu Chu, Jie Wang, and Jian Cao. 2014. Kernel-based reinforcement learning for traffic signal control with adaptive feature selection. In *Proceedings of the 53rd IEEE Conference on Decision and Control (CDC)*. 1277–1282.

[8] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. 2019. Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems* (2019).

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

[10] Peter Dayan and Geoffrey E Hinton. 1993. Feudal reinforcement learning. In *Proceedings of Conference on Advances in Neural Information Processing Systems (NIPS)*. 271–278.

[11] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. 2013. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1140–1150.

[12] Wade Genders and Saiedeh Razavi. 2016. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142* (2016).

[13] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. 2012. Recent development and applications of SUMO-Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements* 5, 3&4 (2012).

[14] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. 2008. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Proceedings of the 19th European Conference on Machine Learning (ECML)*. 656–671.

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

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

[17] Ying Liu, Lei Liu, and Wei-Peng Chen. 2017. Intelligent traffic light control using distributed multi-agent Q learning. In *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems (ITSC)*. 1–8.

[18] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley. 2017. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems* 11, 7 (2017), 417–423.

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

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

[21] Sarvapali D. Ramchurn, Trung Dong Huynh, Feng Wu, Yuki Ikuno, Jack Flann, Luc Moreau, Joel E. Fischer, Wenchao Jiang, Tom Rodden, Edwin Simpson, Steven Reece, Stephen Roberts, and Nicholas R. Jennings. 2016. A Disaster Response System based on Human-Agent Collectives. *Journal of Artificial Intelligence Research (JAIR)* 57 (2016), 661–708.

[22] Thomas L Thorpe. 1997. Vehicle traffic light control using sarsa. In *Online]. Available: citeseer. ist. psu. edu/thorpe97vehicle. html*.

[23] Elise Van der Pol and Frans A Oliehoek. 2016. Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)* (2016).

[24] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 3540–3549.

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

[26] MA Wiering. 2000. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*. 1151–1158.

[27] MA Wiering, J van Veenen, Jilles Vreeken, and Arne Koopman. 2004. Intelligent traffic light control. (2004).

[28] Feng Wu, Sarvapali D. Ramchurn, and Xiaoping Chen. 2016. Coordinating Human-UAV Teams in Disaster Response. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. 524–530.

[29] Feng Wu, Sarvapali D. Ramchurn, Wenchao Jiang, Joel E. Fischer, Tom Rodden, and Nicholas R. Jennings. 2015. Agile Planning for Real-World Disaster Response. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*. 132–138.

[30] Feng Zhu, HM Abdul Aziz, Xinwu Qian, and Satish V Ukkusuri. 2015. A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multi-agent framework. *Transportation Research Part C: Emerging Technologies* 58 (2015), 487–501.