Learning to Coordinate Traffic Signals With Adaptive Network Partition

Jinming Ma and Feng Wu

***Abstract*— Multi-intersection traffic signal control (TSC) is an active research field in multi-agent systems, where traffic signals for each intersection, controlled by an agent, must coordinate to optimize traffic flow. To encourage global coordination, previous work partitions the traffic network into several regions and learns policies for agents in a feudal structure. However, static network partition fails to adapt to dynamic traffic flow, which changes frequently over time. To address this, we propose a novel multi-agent reinforcement learning approach with adaptive network partition. Specifically, we partition the network into several regions according to the dynamic traffic flow over time. To do this, we propose two approaches: one is directly to use graphic neural network (GNN) to generate the network partition, and the other is to use Monte-Carlo tree search (MCTS) to find the best partition with criteria computed by GNN. Then, we design a variant of Qmix using GNN to handle various dimensions of input, given by the dynamic network partition. Finally, we use a feudal hierarchy to manage agents in each partition and promote global cooperation. By doing so, agents are able to adapt to the traffic flow as required in practice. We empirically evaluate our method both in a synthetic traffic grid and real-world traffic networks of three cities, widely used in the literature. The experimental results confirm that our method achieved better performance both in a synthetic traffic grid and real-world traffic networks of three cities, in terms of average travel time and queue length, than several leading TSC baselines.**

***Index Terms*— Adaptive traffic signal control, reinforcement learning, multi-agent coordination, network partition, Monte-Carlo tree search, graphic neural network.**

I. Introduction

NOWADAYS, the growing traffic congestion has brought serious negative impacts on environmental protection, urban economic development and our daily lives. To meet the increasing traffic demand, many efforts have been made to optimize traffic control and improve road capacity. Traditional *traffic signal control* (TSC) mostly relies on pre-defined rules and fixed timing strategy, which cycles the established settings periodically [1]. However, those methods, which heavily depend on expert knowledge and heuristic assumptions, have many weak points. For example, they often cause long traffic delays and are unable to make flexible adjustments based on real-time traffic information. In recent years, researchers [2], [3], [4] have tried to use *deep reinforcement learning* (DRL) for TSC and shown better performance for TSC than traditional methods.

To date, there are several DRL-based methods [5], [6] that control each intersection independently and have no coordination with others. Indeed, the lack of coordination among intersections will lead to poor efficiency of the overall traffic flow. With regards to this, many studies [2], [4] propose to solve TSC using multi-agent RL, which trains agents to cooperate in a centralized or decentralized manner. Generally, decentralized methods have better *scalability* than centralized approaches as each agent independently learns its own policy. However, due to partial knowledge of each agent, decentralized methods may get stuck in *local optima* more easily.

To address this, researchers have investigated several techniques to approach global optima. The most common way is to share observation and fingerprints between adjacent agents for stable cooperative control [4]. Some work uses graph attention networks to facilitate additional information from neighboring intersections to optimize the traffic [3]. Others [7], [8] are based on max-pressure theory to implicitly encourage the cooperation between neighboring intersections. Most recently, FMA2C [9] was proposed to extend MA2C [4] with a feudal structure and improve global coordination among agents. Specifically, it first splits the traffic network into several regions and is fixed after it. Then, a manager-worker hierarchy is constructed, where each region is assigned to a manager and the intersections in the region are controlled by its workers. In this way, managers can cooperate more globally at the regional level and guide the coordination of their workers. However, a major limitation of FMA2C is that the regions are split manually and fixed afterward. As shown later in our experiments, it will become inefficient when the pattern of traffic flow changes frequently. This is a critical drawback for applying it to the real world because the traffic flow may vary during different periods of time (e.g., rush hour, working days, weekends, holidays, etc.). Generally, it is challenging for DRL-based methods to handle such regional changes of traffic systems without re-training the agents.

Against this background, we propose a novel MARL with adaptive network partition for multi-intersection TSC. Specifically, given the underlying traffic network, we first construct the dynamic flow network to model the traffic flow at different periods. Then, we introduce two network partition methods

Manuscript received 13 June 2022; revised 18 February 2023 and 5 May 2023; accepted 15 August 2023. This work was supported in part by the Major Research Plan of the National Natural Science Foundation of China under Grant 92048301 and in part by the Anhui Provincial Natural Science Foundation under Grant 2208085MF172. The Associate Editor for this article was S. Sun. *(Corresponding author: Feng Wu.)*

The authors are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: jinmingm@mail.ustc.edu.cn; wufeng02@ustc.edu.cn).

Digital Object Identifier 10.1109/TITS.2023.3308594

based on graphical neural network (GNN) and Monte-Carlo tree search (MCTS) respectively. For the former, we directly use GNN to generate the network partition according to the traffic flow. This approach is easy to implement and can be trained in an end-to-end manner. However, it is hard to train and may generate unreasonable partition for a large network. In MCTS, we find the best partition through an explicit search process and can scale to a large network. MCTS, as an anytime algorithm [10], can provide the current best solution under time constraints or converge to the optimal solution given sufficient time. Note that the network partition is dynamically updated over time to fit the dynamic flow network. To facilitate different network partitions, we design a variant of Qmix [11] to handle various dimensions of input. Here, we use a feudal hierarchy where manager agents cooperate at the high-level and communicate their sub-goals to lower-level worker agents in the region. In the experiments, we tested our algorithm both in a synthetic traffic grid and real-world traffic networks of three cities. Our experimental results show that our method benefits from the adaptive network partition and significantly outperforms several state-of-the-art methods in terms of average travel time and queue length.

II. Related Work

Here, we briefly review the methods related to our work.

1) Conventional Traffic Signal Control: In conventional control, most methods depend on hand-crafted rules [1], [12]. Fixed time [1] uses a pre-determined cycle signal plan, widely used when traffic flow is stable. SOTL [12] defines the upper and lower limit time of each phase. Reference [1] coordinates traffic signals by modifying the offset between continuous intersections and requires the intersections to have the same cycle length. In fact, it is not easy to manually designed traffic signal plans or rules. To solve difficult to manually design rules, max-pressure [13] aims to balance the queue length between adjacent intersections by minimizing the "pressure" of the intersection. Overall, these approaches still rely on assumptions to simplify traffic conditions and do not guarantee optimal real-world results.

2) RL-Based Traffic Signal Control: RL has been used for TSC so that the control strategy can be adaptively created based on the current traffic condition. Some studies [6], [14] rely on independent Q-learning (IQL). Obviously, they can't optimize the overall objective because they ignore the interaction between adjacent intersections in the road network. For coordinating agents in scenarios of TSC, one typical approach is to train a central agent to control all intersections [5], or jointly model the action among learning agents with centralized optimization [2]. Unfortunately, due to the curse of dimensionality, these *centralized* methods usually have the *scalability* issue and are hard to apply on large road networks.

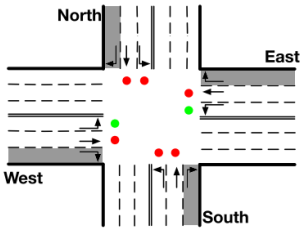
Other studies aim to encourage agents to learn cooperation with others in a decentralized way by adding information from neighbor [3], [4], [15] or design a mechanism that induces cooperation [7], [16]. Some methods apply model-based IQL to each intersection [17] and improved the observability of IQL by neighborhood information sharing [15]. MA2C [4] uses multi-agent A2C to cooperatively control multi-intersections, and it includes information of neighborhood and spatial discount to stabilize the training process. CoLight [3] uses graph attentional networks to facilitate communication between multi-intersections. PressLight [7] uses the theory in max pressure, and designs the pressure as the reward of the agents. Due to the partial observation, these methods may be difficult to achieve global optimal control. Recently, the methods [9], [18] utilize the divide-et-impera framework to divide the whole system into smaller parts and achieve local optimization of each part. Among them, FMA2C [9] combines MA2C with the feudal hierarchy to improve global coordination among agents. We build our algorithm based on FMA2C and make adaptive network partition to adapt to more flexible and complex traffic.

3) Graph Neural Networks: Our proposed method is also related to recent advances of Graph Neural Network (GNN) [19], [20]. GNNs are connectivity models that capture graph dependencies through information transfer between graph nodes [21], [22]. In recent years, a wide variety of graph neural network (GNN) models have been proposed, including convolutional graph neural networks (ConvGNNs) [21], [22], [23], recurrent neural networks (RecGNNs) [24], [25] and graph autoencoders (GAEs) [26], [27]. The general framework of Graph Networks [28] is proposed to support relational reasoning and combinatorial generalization. Graph pooling modules [23], [29], [30] are mainly used to generate graph-level representation based on node representations. Among them, DiffPool [23] is a differentiable graph pooling module that can generate hierarchical representations of the entire graph and learn a differentiable assignment mapping nodes to a set of clusters based on their learned embeddings.

Recently, many works incorporate GNNs into the traffic domain to capture the spatial dependency, due to the traffic data are graph-structured. As summarized in the survey [31], [32], existing works focus on incorporate GNNs into: Traffic forecast, Traffic signal control and Trajectory prediction. In traffic forecast, the main focus is on traffic flow forecast [33], [34], traffic speed forecast [35], [36] and traffic demand forecast [37], [38], [39]. In addition, some traffic signal control works [40], [41] consider that the multi-intersection traffic lights is spatially and temporally influenced. And they propose methods to control traffic lights by effectively capturing the spatio-temporal dependency with GNN-based deep learning. Trajectory prediction [42], [43] mainly studies the accurate prediction of pedestrian trajectory and vehicle trajectory, which can play important role in the downstream tasks. Beside them, we are the first use GNN for partitioning regions beneficial in RL-based traffic signal control.

In our paper, we use DiffPool [23] to learn how to cluster nodes together to build a hierarchical structure on top of the underlying road network, i.e., integrating the information of the entire road network to obtain the optimal network partition. Meanwhile, the hierarchical structure generated by DiffPool can be naturally combined with the feudal structure. In contrast, most of the GNNs are to generate embeddings for all the nodes in the graph and then to globally pool all these node embeddings together, which ignores the hierarchical

(a) Multi-intersection traffic network in Hangzhou, China.



(b) An intersection and its 12 movements.

Fig. 1. Example of the traffic network and intersection.

structure of the graph. Thus, DiffPool is an ideal technique for our approach.

III. Background

In this article, we consider the TSC problem in a multi-intersection network as illustrated in Figure 1(a). Here, each intersection has 12 traffic movements, consisting of four approaches and three lanes (turning right, turning left and going straight) per approach. As shown in Figure 1(b), the green dot indicates that the movement is allowed and several non-conflicting movements can be combined in a phase. The objective of TSC is to choose the optimal phase of each intersection to maximize the traffic flow. As in the literature, this problem can be modeled as a Markov game, where each intersection in the traffic network is controlled by an agent.

A. Markov Game for Traffic Signal Control

Specifically, the Markov game is defined by a tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$. $\mathcal{S}$ is a finite set of states, and each agent can observe a part of the state $s \in \mathcal{S}$ as its observation $o \in \mathcal{O}$. $\mathcal{A}$ is a set of actions for each agent. $\mathcal{T}$ is the transition function with joint action of all agents $\mathbf{a}$. Each agent obtains an immediate reward r by a reward function $\mathcal{R}(s, \mathbf{a})$.

For TSC, each agent observes the quantitative descriptions (i.e. observation) of its intersection, such as queue length, waiting time and delay. Then, the agent chooses a phase (i.e. action) for the intersection and receives an immediate reward r from the environment, which indicates the traffic situation. The goal is to optimize the situation of the overall traffic network. In the literature of TSC, *average travel time* of all vehicles in a road network is the most frequently used measure of the overall traffic network [44]. Specifically, average travel time is defined as the average time difference between the time when all vehicles enter the network and the time when they leave the network. Compared with the conventional TSC methods that heavily rely on pre-defined rules, the main advantage of solving TSC by MARL is that agents can learn a better policy by directly interacting with the dynamic environment.

B. Feudal Multi-Agent Advantage Actor-Critic

Feudal Multi-agent Advantage Actor-Critic (FMA2C) [9] is an extension of MA2C with feudal hierarchy for regional coordination among regions. For traffic signal control, the large-scale networked system is firstly divided into multiple regions, and the hierarchical structure of Manager-Worker is constructed. A manager agent is assigned to each of these regions, and each intersection in the region is controlled by a separate worker agent. In more detail, each manager coordinates with other managers for maximizing the long-term, global reward, and learns to communicate sub-goals to multiple simultaneously operating workers. By receiving the goal from its manager, each worker needs to achieve the managerial goal according to the completion of the sub-goals, in addition to maximizing its own local rewards that they experience from the environment. By doing so, managers can have global cooperation at the regional level and manage the coordinated control of workers within their regions.

1) Learning Policy of Manager: For manager, FMA2C learns the policy by MA2C [4]. This method combines value-based (such as Q learning) and policy-based (such as Policy gradients), and includes two networks in each agent's model: actor network and critic network. The actor network is used to generate behavior policies, and the critic network is used to evaluate the behavior policies generated by the actor. Meanwhile, the actor network improve their behavioral policies based on the evaluation of the critic network.

Given the mini-batch B^M that contains the experience trajectories generated by the manager's interaction with the environment. The critic network is updated by the loss:

$$\mathcal{L}(w^M) = \frac{1}{2|B|} \sum_{t \in B^M} (R_t^M - V_w^M(s_t^M))^2 \tag{1}$$

where $R_t^M = \sum_{\tau=t}^{T-1} \gamma^{\tau-t} r_\tau^M + \gamma^{T-t} V_{w^-}(s_t^M)$ is the estimating local returns and $V_w^M(s_t^M)$ is the approximate state values.

Given the advantage value $A_t^M = R_t^M - V_{w^-}(s_t^M))$, each manager minimizes the actor loss to update its parameter θ^M:

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

2) Learning Policy of Worker: As aforementioned, worker learn policy that fulfill both the managerial goals and its local objectives. Therefore, the state of each worker consists of the local observations and the subgoal communicated by its manager, i.e., $s^W = [o^W, g]$. Meanwhile, the reward of each worker is augmented by the managerial reward which considering the subgoal g as:

$$\hat{r}_t^W = r_t^W + \sigma(o_t^W, g_t) \tag{3}$$

where r_t^W is the intrinsic reward of the worker and σ is a function mapping from the worker's observation and the subgoal to a real number. In FMA2C, they use the value of the angle between the motion vector of the observation and the subgoal direction vector to measure the degree of following with the subgoal: $\sigma(o_t^W, g_t) = d_{cos}(o_{t-1}^W - o_t^W, g_t)$.

Similarly, the critic network of worker can update their parameter w^W with each mini-batch B^W as follow:

$$\mathcal{L}(w^W) = \frac{1}{2|B|} \sum_{t \in B^W} (R_t^W - V_w^W(s_t^W))^2 \tag{4}$$

where $R_t^W = \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \hat{r}_\tau^W + \gamma^{T-t} V_{w^-}(s_t^W)$ is the estimating returns and $V_w^W(s_t^W)$ is the approximate state values.

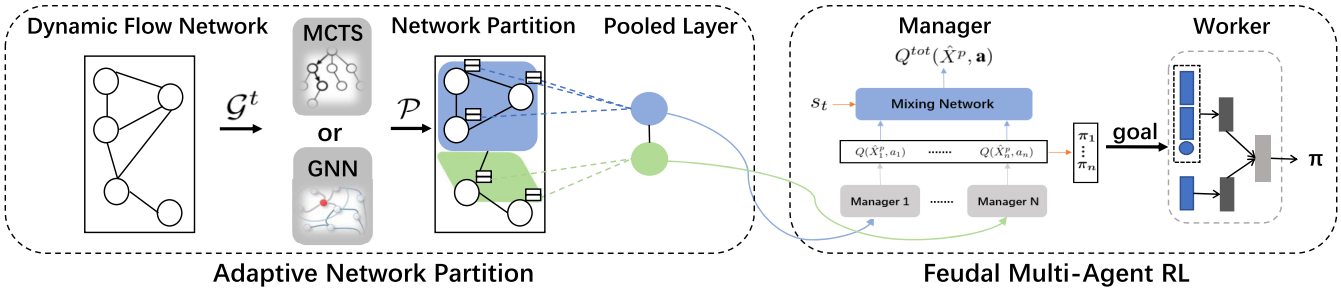


Fig. 2. Overall framework of Adaptive Feudal Multi-agent Reinforcement Learning (AFMRL).

Now, each worker minimizes the policy loss to update its parameter θ^W as:

$$\mathcal{L}(\theta^W) = \frac{1}{2|B|} \sum_{t \in B^W} log\pi_\theta(a_t^W | s_t^W) A_t^W \tag{5}$$

where the advantage value is $A_t^W = R_t^W - V_{w^-}(s_t^W))$.

Although the feudal hierarchy does improve global coordination among multi-intersections, the network partition (i.e., which region is controlled by a manager) is *static* and not flexible when the traffic pattern changes frequently. As we observed in the experiments, when traffic patterns change, the hierarchy becomes less effective if the congested area spans two static regions. To address this, we propose a novel RL approach with *adaptive* network partition that is more flexible and can adapt to complex traffic patterns.

IV. Method

In this section, we propose AFMRL | Adaptive Feudal Multi-agent Reinforcement Learning with dynamic network partition for TSC. The overall framework of our method is shown in Figure 2. From left to right, we start with a formal definition of dynamic flow network and introduce two adaptive network partition methods based on GNN and MCTS respectively. For the former, we directly use GNN to generate the network partition. For the latter, we use MCTS to search the best network partition whose value is measured by GNN. Note that the network partition is done over time to fit the dynamic traffic flow due to the same traffic pattern usually lasts for a period of time and does not change very quickly in the real-world (e.g., morning-evening rush hours). Then, we designed a variant of Qmix to process different dimensional regions given the network partition. Finally, we construct a feudal hierarchy to learn policies for the regions and each intersection.

A. Adaptive Network Partition

We model a traffic network as directed graph $G(\mathbb{V}, \mathbb{E})$, where each vertex $v \in \mathbb{V}$ represents an intersection and each edge $e_{ij} \in \mathbb{E}$ represents the road from intersection i to j. Now, we introduce the dynamic flow network to capture the dynamic traffic flow that changes over time. Specifically, the *dynamic flow network* is defined as $\mathcal{G}^t(G, F^t)$, where: G is the traffic network; $f_{ij}^t = (e_{ij}, w) \in F^t$ is the traffic flow in edge e_{ij} with traffic density descriptions w, which is ratio of queue length over the capacity of the road.

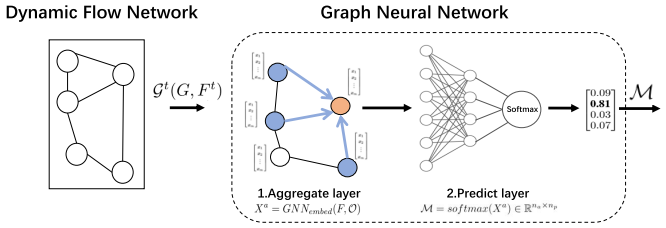


Fig. 3. Illustration of the GNN for generating network partition $\mathcal{P}$ (The colored arrow line represents the process that aggregate feature information from neighbors).

Given the dynamic flow network, we partition the network $\mathcal{G}^t$ into m disjoint regions $\mathcal{P} = \{P_1, \ldots, P_m\}$, where $\forall_{P_i, P_j}, \cup_{k=1}^m P_k = \mathcal{G}, P_i \cap P_j = \emptyset$. Here, every T timesteps, the dynamic flow network can be partitioned into different disjoint regions depending on the real-time traffic flow. As aforementioned, our goal is to adaptively partition the network so that it is beneficial for the learning algorithm.

To this end, we propose two approaches: one is to directly generate the network partition through GNN, and the other is using MCTS to search the network partition with criteria computed by a separate GNN. The advantage of the former is that it can train the entire network in an end-to-end fashion. However, it may fail to generate a reasonable partition for a larger network. On the contrary, the latter can generate a good network partition through explicit search in a feasible time. We will describe both methods in the following sections and compare their performance in the experiments.

1) GNN for Network Partition: Here, we introduce our approach by applying the GNN to generate the network partition $\mathcal{P}$. As shown in Figure 3, the GNN takes the dynamic flow network $\mathcal{G}^t(G, F^t)$ as input, which contains the information of traffic network G and the real-time flow F^t. Inspired by [23], the first step is to generate embedding of nodes and aggregate feature information from the neighbors (blue) to the target node (orange), which is named aggregate layer. Then we input the embedding of the target node into the predict layer with $softmax$ to get the assignment vector, which can be regarded as a multi-classifier. The output dimension of the GNN corresponds to the number of regions. All in all, we compute the assignment matrix $\mathcal{M}$ of agents using a *embed* GNN that takes the input observation of intersections $\mathcal{O}$ and the real-time flow $F^t \in \mathbb{R}^{n_a \times n_a}$ as follow:

$$\mathcal{M} = softmax(GNN_{embed}(F, \mathcal{O})) \in \mathbb{R}^{n_a \times n_p} \tag{6}$$

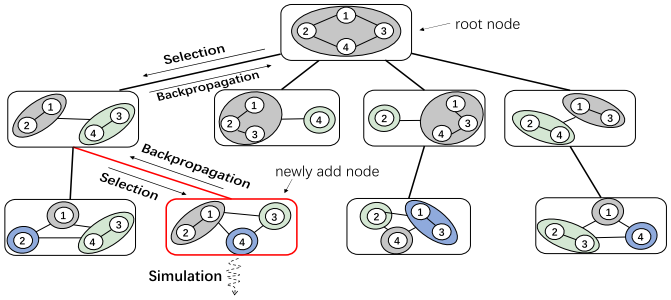


Fig. 4. Partial search tree built by MCTS for network partition.

where n_a is the number of intersections, n_p is the number of regions. Then, we can get the network partition $\mathcal{P}$ through the assignment matrix $\mathcal{M}$, i.e., region P_i consists of the intersections whose i-th column of the matrix is maximum.

Our objective of training the GNN is to generate a network partition with a good collective payoff that the agents can gain by forming the network partition in TSC. In practice, it is generally difficult to train the GNN using only gradient signal from the RL in an end-to-end fashion as shown in Figure 2. Hence, we use an auxiliary link loss to encode nearby nodes that should be pooled together as suggested by [23], i.e., minimizing the loss $\mathcal{L}_{aux} = \|A, \mathcal{M}\mathcal{M}^T\|$, where $\|\|$ is the Frobenius norm and A is the adjacency matrix of the traffic network. In summary, we train the GNN in an end-to-end fashion with the Loss function $\mathcal{L} = \mathcal{L}_{aux} + \mathcal{L}_{Qmix}$.

Even with this improvement, we observed that GNN may not provide the best network partition, because the complex topological structure of graphs is hard to learn with limited feedback. It becomes more severe on a larger traffic network, which will be shown in our experiments. This motivates us to propose an alternative approach based on MCTS.

2) MCTS for Network Partition: Now, we turn to propose our MCTS method to search for the best network partition in the dynamic flow network. Inspired by [45], we model the partition problem as a search process over tree nodes. As shown in Figure 4, each tree node is associated with a partition and the root node of the search tree is the original network containing all agents, e.g., $\mathcal{P} = \{\{1, \ldots, n\}\}$. The children of each node are obtained by bipartition of one region of its parent, e.g., a child of the root node is $\{\{1, \ldots\}, \{\ldots, n\}\}$. Each branch of the tree expands until the termination condition meets at a leaf node. When all branch reaches their leaf, the search is completed.

We incrementally build the search tree iteration by iteration. In each iteration, as shown in Figure 4, we start the search from the entire network down to split network and expand the search tree based on four steps of MCTS. In the *selection* step, an optimal child node is selected among all children so that the tree can expands to the subspace where the optimal solution is most promising. Similar to previous MCTS methods, we use the UCB1 heuristic [46] to select branches as:

$$UCB1(\mathcal{P}') = V'(\mathcal{P}') + c\sqrt{logN(\mathcal{P})/N(\mathcal{P}')} \tag{7}$$

where $V'(\mathcal{P}) = max_{\mathcal{P}' \in Tree'(\mathcal{P})} V(\mathcal{P}')$ is the current maximal value with the current search space $Tree'(\mathcal{P})$, N($\mathcal{P}$) is the frequency that the node $\mathcal{P}$ is visited when searching, and c is a constant parameter. Given this, we can select the child node that maximizes the UCB1($\mathcal{P}'$), i.e., $\mathcal{P}^* = \arg\max_{\mathcal{P}' \in Child(\mathcal{P})} UCB1(\mathcal{P}')$. *In the UCB1, the node value is augmented by an exploration bonus that is higher for rarely tried children*. Intuitively, if $N(\mathcal{P}')$ is equal for all children, it will select the child with the maximal $V'(\mathcal{P}')$ because the remaining term is equal for all of them. However, if some child is much less frequently visited than others, the term $\sqrt{logN(\mathcal{P})/N(\mathcal{P}')}$ will become significant and bias the selection towards it. By using the UCB1, MCTS has the *anytime* property and can usually find the best partition much earlier before checking all the nodes. In the *expansion* step, if the selected child $\mathcal{P}^* \in Child(\mathcal{P})$ is currently not in the tree, we expand by adding a new node $\mathcal{P}^*$ as a child of $\mathcal{P}$.

In the *simulation* step, to evaluate the value $V'(\mathcal{P}^*)$ of the new node $\mathcal{P}^*$, we use the default policy to perform a rollout search by successively bipartitioning a region until the termination condition is met. The termination condition is met when only partitions with the pre-defined smallest size (not singleton) are left. Note that we aim to coordinate the traffic regions and such regions (e.g., city districts) are not very small in real-world scenarios. Therefore, we can terminate a branch when the partition becomes too small. Then, the value of $V'(\mathcal{P}^*)$ is initialized by $V'(\mathcal{P}^*) = max_{\mathcal{P}' \in Trace(\mathcal{P}^*)} UCB1(\mathcal{P}')$, where $Trace(\mathcal{P}^*)$ is a set of partitions encountered during the rollout search from $\mathcal{P}^*$. In rollout search, for $\mathcal{P}$, the default policy to perform network partition is to select the partition $\mathcal{P}' \in Child(\mathcal{P})$ with maximizing the value $V(\mathcal{P}')$.

In the *backpropagation* step, after the simulation of $\mathcal{P}^*$, all of its ancestors are updated by backpropagating the value $V'(\mathcal{P}^*)$. For each ancestor, its value is updated by $V'(\mathcal{P}^*)$, which is the optimal solution in the sub-tree rooted by it.

Each iteration expands the search tree, and as the number of iterations increases, the size of the search tree continues to increase. When all nodes are added to the tree or time runs out, the complete solution space has been searched. In the former case, we can search the optimal network partition while in the latter case the currently best solution is searched.

3) Network Partition Value: In MCTS, one remaining challenge is to define the value function $\mathcal{V}(\mathcal{P})$ of $\mathcal{P}$ for any network partition $\mathcal{P}$. With this, the goal of MCTS is to find the most valuable partition $\mathcal{P}^*$ in the set of feasible partition denoted by $\mathbb{P}$, i.e., $\mathcal{P}^* = \arg\max_{\mathcal{P} \in \mathbb{P}} \mathcal{V}(\mathcal{P})$. Unfortunately, this value function is unknown in our setting and must be learned along with the RL algorithm. Here, we use GNN to extract useful feature embeddings from the input graphs, including vertex, edge and topology information, and predict the network partition value. Note that this is a separate GNN that is structurally different from the aforementioned GNN for network partition.

As shown in Figure 5, the network takes the observation of agents $\mathcal{O}$, the real-time flow F^t and the assignment matrix $\mathcal{M} \in \mathbb{R}^{n_a \times n_p}$ as input. We first generate the embeddings X^a of agents by an *embedding* GNN module [23], which is applied to $\mathcal{O}$ and F as below:

$$X^a = GNN_{embed}(F, \mathcal{O}) \in \mathbb{R}^{n_a \times d} \tag{8}$$

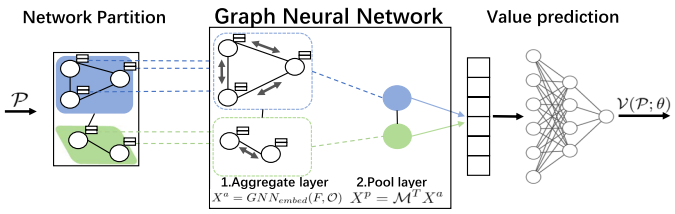


Fig. 5. Illustration of predicting the network partition value $\mathcal{V}(\mathcal{P};\theta)$ by GNN. The colored rectangle represents a region, and the black line represents the traffic flow between them. The colored dotted line represents the process according to Equations 9 and 10.

where d is the dimension of agents' embeddings. In this step, the GNN module performs relation reasoning that propagates information across edges of the graph. Then, we generate a coarsened adjacency matrix $\mathcal{F}$ denoting the connectivity strength between each pair of regions (i.e., the traffic flow between regions), and the embeddings X^p for regions by pooling these agents' embeddings in the region together according to the assignment matrix $\mathcal{M}$:

$$\mathcal{F} = \mathcal{M}^T F \mathcal{M} \in \mathbb{R}^{n_p \times n_p} \tag{9}$$

$$X^p = \mathcal{M}^T X^a \in \mathbb{R}^{n_p \times d} \tag{10}$$

Similarly, we take the relational matrix between the regions $\mathcal{F}$ and their embeddings X^p through an *embedding* GNN module to get new embeddings $\hat{X}^p$ for the regions:

$$\hat{X}^p = GNN_{embed}(\mathcal{F}, X^p) \in \mathbb{R}^{n_a \times d} \tag{11}$$

Finally, the new embeddings feed in the neural network to output the value $\mathcal{V}(\mathcal{P};\theta)$.

In the training process of the GNN, we expect to fit $\mathcal{V}(\mathcal{P};\theta)$ as the collective payoff that the agents can gain by forming the network partition $\mathcal{P}$ in TSC. Therefore, we use the rewards r, which is received from the environment by interacting with the environment during T steps, as the training signals. Since the high-level reward r represents the local travel time of the traffic network, it can be used to represent how much collective payoff that the agents can gain by forming the network partition $\mathcal{P}$. To put together, the network is trained in an end-to-end fashion by minimizing $\mathcal{L} = \text{MSE}(\mathcal{V}(\mathcal{P};\theta), r)$.

Since the GNN requires a learning process, its prediction value may not be useful for MCTS especially in the early stage. Note that the learning process of GNN and the search process of MCTS are inter-dependent: GNN needs good partitions to learn their values while MCTS requires accurate values to find the good partition. Therefore, we introduce a heuristic function to boost the learning process of GNN.

We borrow ideas from graph theory [47] to evaluate how strong the connection is between two regions: $cut(P_i, P_j) = \sum_{u \in P_i, v \in P_j} f(u, v)$, where $f(u, v)$ is the traffic flow between intersections u and v. Intuitively, a good partition should minimize this value between every pair of regions. Furthermore, to avoid undesirable bias for partitioning out small sets, we use "Normalized cut" [48] to measure the disassociation of the network partition:

$$Ncut(P_1, \ldots, P_m) = \sum_{i,j=1, i \neq j}^{m} \frac{cut(P_i, P_j)}{assoc(P_i)} \tag{12}$$

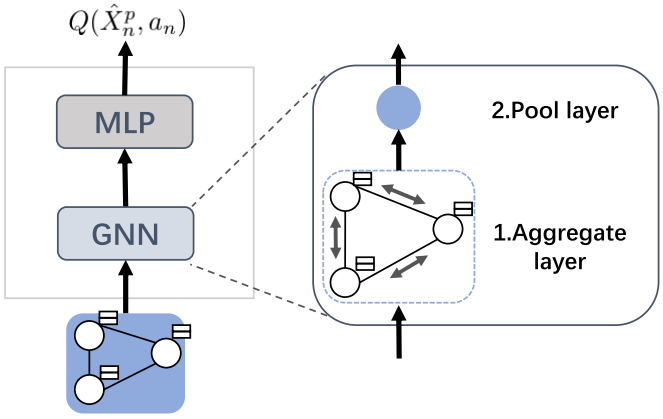


Fig. 6. Illustration of the GNN for handle the various dimensions of independent Q-value network in Qmix.

where $assoc(P_i) = \sum_{u \in P_i, v \in \mathbb{V}} f(u, v)$ is the total connection from nodes in P^i to all nodes in $\mathcal{G}$. By minimizing $Ncut$, we will find the apposite network partition where the connection within each region is high while the connection between each region is low. Intuitively, this meets our goal of partitioning the traffic networks.

Given this, we combine the heuristic with the prediction value and compute the value of a network partition $V(\mathcal{P})$ as:

$$V(\mathcal{P}) = (\alpha - 1) Ncut(\mathcal{P}) + \alpha \mathcal{V}(\mathcal{P}) \tag{13}$$

where α is a adaptive weighting coefficient, which is low in the early stage of training and gradually grows along with the learning process.

B. Feudal Multi-Agent Reinforcement Learning

Here, we follow the FMA2C framework [9] and form a feudal hierarchy with managers and workers, where a manager controls a region in the partition and the agents in the region are its workers who directly control the traffic intersection. In more detail, the manager is tasked with maximizing the long-term, global reward, and learns to communicate sub-goals to multiple simultaneously operating workers. The workers need to learn how to act based on the managerial reward according to the completion of the sub-goals, in addition to maximizing immediate rewards that they experience from the environment. Similar to FMA2C [9], we train the policies of workers by the MA2C algorithm.

1) The Variant of Qmix for Managers: As aforementioned, each manager controls a region and coordinates the workers inside the region. Note that our objective is to optimize the traffic flow of the entire network. Therefore, the managers of different regions also need to cooperate with each other. To achieve this, we employ Qmix [11] to learn managers' policies. Notice that, for the different situations of the traffic network, there will be different network partitions, i.e., the number of regions and the number of agents in the region. Therefore, Qmix must be modified to handle this issue.

As shown in Figure 6, we use a GNN to handle the dynamic dimensions of input, which is a separate GNN with the same structure as shown in Figure 5. In more detail, we compute

the feature embeddings $\hat{X}^p$ of regions by Equations 8-11, where Equation 11 stabilizes the training process by one-step relational reasoning. Note that, for empty regions, their feature embeddings are $\mathbf{0}$.

Given the GNN, we use the m-th row of $\hat{X}^p$ as the feature of the region m and feed it into the Q-value network Q. By doing so, we can use a unified Q-value network to process dynamic dimensions of input and output the individual Q-value function $< Q^1, \ldots, Q^m >$, rather than train multiple models for different network partitions and switch the model according to different network partitions.

For the mixing network, we still adopt the method introduced in Qmix [11]. The mixing network takes the environment state s as input and generates the non-negative weights $w(s) \geq 0$ and bias $b(s)$ of the layer in the mixing network. The joint action-value network Q^{tot} can be represented as $Q^{tot} = \sum_{i=0}^{m} w^i(s) Q^i + b(s)$. The network is trained in an end-to-end manner by using the loss:

$$\mathcal{L}(\theta) = \frac{1}{2|B|} \sum_{t}^{|B|} (y_t^{tot} - Q^{tot}(\hat{X}^p, \mathbf{a}, s; \theta))^2 \qquad (14)$$

where B is the transitions sampled from the replay buffer, $y^{tot} = r + \gamma max_{\mathbf{a}'} Q^{tot}(\hat{X^p}', \mathbf{a}', s'; \theta^-)$ and θ^- are the parameters of a target network as in DQN.

V. Experiments

We implemented the experiments using the CityFlow simulator [49] | a MARL environment for large-scale city traffic scenarios. To evaluate the performance of our method, we compared with several conventional and state-of-the-art RL methods for traffic signal control in a synthetic traffic grid and three real-world traffic networks.[1]

A. Model Definition

Here, we introduce the observation, action, and reward definition for the feudal structure, i.e., manager-worker hierarchy. There are many different definitions in the literature [44]. In this paper, we refer to the definitions of [9], with the only difference of adjustment in the manager settings.

1) Observation: For worker i, the observed traffic information is some quantitative descriptions of the intersection i, i.e., queue length, waiting time and delay. In this paper, we choose queue length of all lane in the intersection as the observation, $o_{t,i}^W = \langle q_1, \cdots, q_{l_n} \rangle$, where l_n is the number of lanes in intersection i. For manager k, the observation is the traffic flow in the region k, i.e., $o_{t,k}^M = \langle \ldots, o_{t,i}^W, \ldots \rangle$, where $i \in Region_k$. And the region's observation will be input into a GNN to extract the its feature emmbddings.

2) Action: For each worker, it decides which phase to be selected from a phase set. In other words, the action is the index of a phase. For each manager, it sets a sub-goal for its region. And we consider the sub-goal as a possible traffic flow, which is a combination of $[N, E, W, S]$ traffic flows, e.g, *north-south* and *east-west* traffic flows.

[1]https://traffic-signal-control.github.io

3) Reward: For worker i, the reward received from environment is the queue length of all lanes in the intersection i, which can be represented as $r_i = -\sum_{l=1}^{l_n} q_l$. For manager k, we define a reward of long-time horizon. Note that we employ Qmix [11] to estimate joint action-values as a complex non-linear combination of per-manager values that promote the same optimization objective of managers. We consider the local travel time, which is the sum of local travel time of all intersections of the traffic network. The local travel time of an intersection is the time discrepancy between entering and leaving the local area of the intersection. Therefore, the manager can indirectly optimize the average travel time by optimizing the local travel time.

B. Datasets

The road networks are illustrated in Figure 7. The detail setting of the road networks are as follow: $D_{4\times4}$: The road network contains 16 intersections in a 4×4 grid. The traffic volume is randomly sampled from a Gaussian distribution with a mean of 500 vehicles/hour/lane. $D_{Jinan,Hangzhou}$: The road network of Jinan and Hangzhou contains 12 and 16 intersections in a 4×3 and 4×4 city network, respectively. The traffic flow is generated from surveillance camera data. $D_{Manhattan}$: The road network of Manhattan contains 48 intersections in a 16×3 city network. The number of vehicles generated is sampled from taxi trajectory data. Specifically, we use the synthetic grid to evaluate our method under various flexible traffic patterns and real-world datasets to test the practicality of our method.

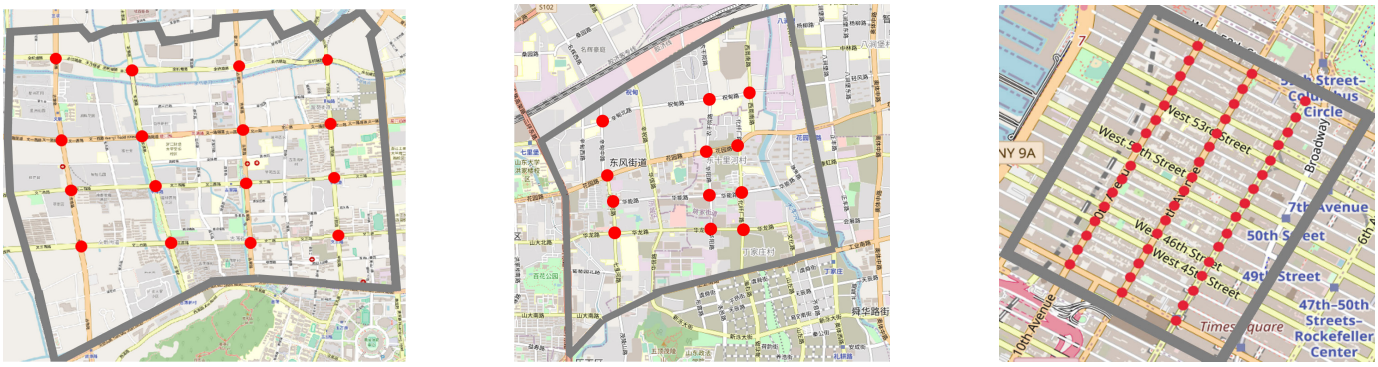
C. Experimental Settings

1) Compared Baselines: We compare our AFMRL with the following conventional and leading RL methods.

- **SOTL** [50] is controlled with demand responsive rules which compare the current phase with current traffic. It is a conventional method that utilizes current traffic.
- **MaxPressure** [13] aims to control the intersection by balancing queue length between neighboring intersections by minimizing the "pressure" of the phases. It is the SOTA conventional method for the network-level TSC.
- **CoLight** [3] uses graph attention network to facilitate information between neighbors.
- **PressLight** [7] uses the max-pressure theory and designs the pressure as the reward of the agents, which has good performance in multi-intersection TSC.
- **MA2C** [4] uses multi-agent A2C to cooperatively control multi-intersections. And it includes information of neighborhood and spatial discount to stabilize the training.
- **FMA2C** [9] is an extension of MA2C with feudal hierarchy for traffic signal control. It achieves better global cooperation through feudal hierarchy.

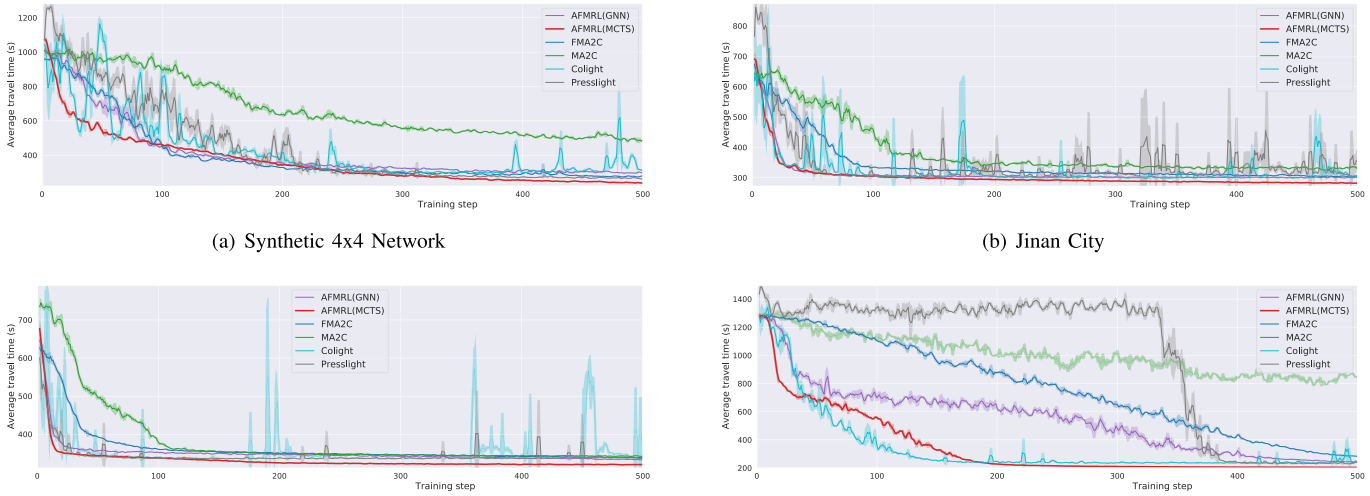
For a fair comparison, we use the same experimental settings for specifying the Markov game model as in the FMA2C paper [9]. All the GNNs were implemented using the network structure in DiffPool [23].

2) Evaluation Metric: There are many different metrics to measure traffic conditions. The most common metrics of traffic signal control are to minimize the *queue length* and *average*



(a) Gudang Sub-district, Hangzhou, China (b) Dongfeng Sub-district, Jinan, China (c) Manhattan, New York City, USA

Fig. 7. Illustration of traffic networks in the real-world datasets.



(a) Synthetic 4x4 Network (b) Jinan City

(c) Hangzhou City (d) Manhattan City

Fig. 8. The training curves of average travel time (sec.) in different networks. The solid line and shade are mean and standard deviation respectively.

travel time. Queue length of the intersection is the number of vehicles in all incoming lanes. Average travel time is defined as the average time difference between the time when all vehicles enter the network and the time when they leave the network. In addition, the former is used to assess short-term traffic control, while the latter is used to assess long-term traffic control.

D. Experimental Results

As commonly in the literature, we use the queue length and the average travel time as the metrics to measure short-term and long-term traffic conditions respectively. The queue length is the number of queuing vehicles in the road network. The travel time is the time difference between the time when all cars enter the network and the one when they leave it. The average values were computed by several runs with different random seeds after convergence in training.

1) Training Results: As shown in Figure 8(a-d), our method substantially outperformed all the compared RL methods, in the speed of convergence, the stability of learning, and the quality of policy. By comparing with FMA2C, we can find that our method with adaptive network partition converged faster and got better results in real-world traffic networks. This indicates the advantage of our method in situations where traffic patterns change frequently.

2) Evaluation Results: The performance of all methods in four traffic networks is summarized in Table I. Compared with the RL methods, the traditional methods have relatively poor performance. As aforementioned, traditional methods that rely heavily on pre-defined rules do not work well with dynamic traffic. Among the RL methods, FMA2C gained a slightly better performance due to the global cooperation brought by the feudal structure. All in all, our method achieved the best results especially for the large network (i.e., Manhattan).

As shown in Figure 9, we show a sequence of network partitions over time, which dynamically fit the real-time traffic flow. We can see that the traffic flow changed over time, and the traffic density of the connection between each pair intersections is various at different moments. Due to our adaptive network partition, as expected, the inter-regional connection is relatively low, while the intra-regional connection is relatively high. Intuitively, this is a reasonable network partition and beneficial for learning good policies for the feudal framework. Next, we will further investigate the reasons for the advantage of our method through the ablation experiments.

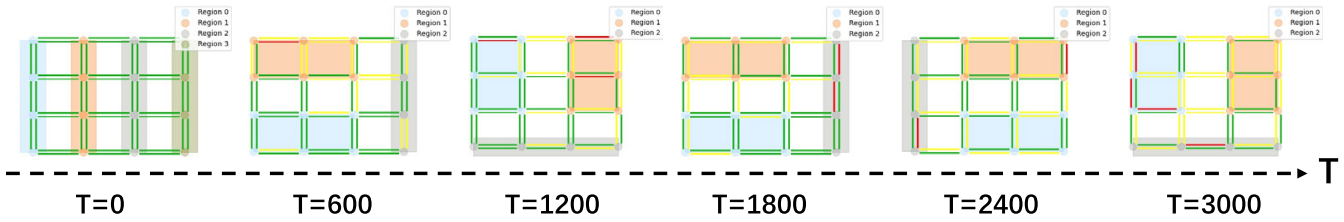


Fig. 9. Illustration of the sequence of network partitions changed dynamically along with the traffic flow in an episode. Each intersection is represented by a vertex, and the connected edges represent the traffic flow density (green, yellow, and red indicate the degree of congestion). Different rectangles represent different regions.

TABLE I

PERFORMANCE IN SYNTHETIC 4 × 4, JINAN, HANGZHOU AND MANHATTAN (BEST VALUES ARE IN BOLD)

Method	Average Travel Time [sec.]				Avg. queue length [#veh.]			
	Synthetic 4 × 4	Jinan	Hangzhou	Manhattan	Synthetic 4 × 4	Jinan	Hangzhou	Manhattan
SOTL	506.72	423.39	619.03	1745.09	76.01	26.71	23.56	88.51
MaxPressure	251.12	337.36	374.78	305.96	27.60	13.96	6.13	8.91
CoLight	275.55	331.59	341.61	238.35	31.87	12.08	3.13	4.19
PressLight	262.28	309.05	336.47	231.21	29.47	9.46	2.89	3.85
MA2C	421.38	328.83	340.96	351.86	53.51	12.73	3.10	5.56
FMA2C	265.34	302.74	334.95	247.91	28.90	10.12	3.03	4.98
AFMRL (GNN)	274.37	299.25	329.52	223.10	31.73	8.77	2.68	3.52
AFMRL (MCTS)	**228.96**	**278.95**	**318.21**	**179.65**	**23.36**	**5.56**	**1.63**	**1.73**

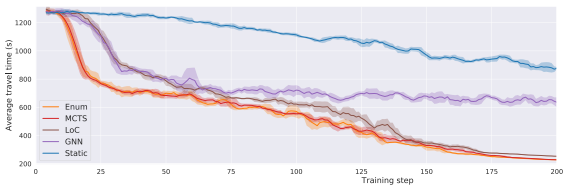
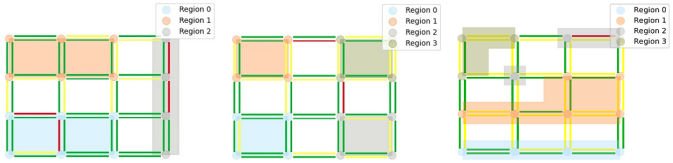


Fig. 10. Ablation results for different network partition methods.



(a) MCTS (b) Static (c) GNN

Fig. 11. Illustration of the network partition generated by different methods on the 4 × 4 grid. Each vertex represent the intersection, and the edges represent the traffic flow density (green, yellow, and red indicate the degree of congestion). Different rectangles represent regions.

E. Ablation Experiments

1) Partition Techniques: Here, we study the usefulness of different network partition techniques. Specifically, we tested the methods with static network partition (Static), GNN for network partition (GNN), and MCTS for network partition (MCTS). Exhaustive enumeration (Enum) for searching the best network partition is also included for reference though it is not scalable for large networks. In addition, we also compared with the graph theory based partition method using *level of connectivity* (LoC) [51].

As shown in Figure 10, the static network partition has the worst performance when it does not match with the dynamic traffic flow. In more detail, when the congested area spans two regions, the manager-level coordination is less effective. As shown in Figure 11(b), the traffic density of the connection between the regions is high at this moment, e.g., the connection between the upper right region with its neighbors. In this case, the intra-regional connection is low, while the inter-regional connection is high. In contrast, GNN does adapt to the dynamic traffic flow. Therefore, its performance is better than the one with static partition. However, it is very hard to train and often fails to produce reasonable partition especially for large networks. For example, as shown in Figure 11(c), the intersections of the same region are not adjacent, which is usually a bad network partition for traffic signal control. Due to LoC measure the similarity between intersections for network partition, LoC can generate reasonable network partition. However, LoC only considers the information of the traffic network, without the feedback from the collective payoff of RL after forming the network partition. As a result, the network partitions generated by LoC are not effective for policy improvement in the FMA2C framework. For MCTS, as shown in Figure 11(a), we can see that the partition is preferable, where the intra-regional connection is high, and the inter-regional connection is low. As a result, MCTS has the best performance compared with the other methods. When compared with the enumeration baseline, MCTS gets very close results to it. Most importantly, MCTS is much more efficient and can scale to large networks.

As aforementioned, MCTS is an anytime algorithm. This property means that it can be stopped at anytime and provide the current best solution. In addition, it is complete and will eventually converge to the global optimal solution if a

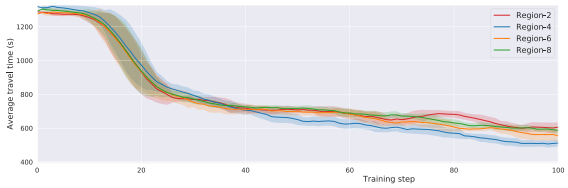
TABLE II
COMPARISON OF DIFFERENT SEARCH TECHNIQUES FOR ADAPTIVE NETWORK PARTITION. MCTS(10) MEANS 10 ITERATIONS

Method	Manhattan City		Hangzhou City	
	N-Cut Score	**Time (s)**	**N-Cut Score**	**Time (s)**
K-Way	0.061	0.09	0.12	0.003
Kernighan-Lin	0.031	0.42	0.12	0.012
MCTS(10^1)	0.497	0.87	0.13	0.015
MCTS(10^2)	0.030	2.56	**0.11**	0.08
MCTS(10^3)	0.023	6.85	**0.11**	0.55
MCTS(10^4)	**0.021**	48.95	**0.11**	5.40
BFS	**0.021**	306.67	**0.11**	0.038

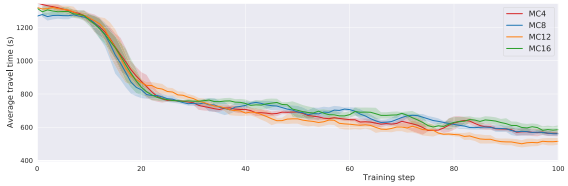
sufficient amount of time is given. In practice, this property is appealing to TSC because the network may be very large and time is limited. To demonstrate this property, we conduct experiments with several network partitioning methods to test the score of partitions and time consumption. According to the recent survey, existing search/optimization methods for network partitioning can be categorized into several types [52], including but not limited to exhaustive enumeration, heuristic search and multilevel methods. On one hand, the exhaustive methods (e.g., best-first search, BFS) can guarantee to find the optimal solution but it is time-consuming and not scalable to solve large problems. On the other hand, the heuristic methods such as Kernighan-Lin [53] and the multilevel methods such as K-Way [54] can return a solution quickly. However, such search methods have no guarantee to find the optimal solution and can easily get trapped in a local optimum [52]. In contrast, MCTS offers a better tradeoff between solution quality and computational time due to its anytime property. In Table II, we test MCTS with the different number of iterations and compare it with several network partitioning methods. As we can see, the solution quality of MCTS improves with the number of iterations increasing, in which the runtime only increases linearly. With 10^4 iterations, MCTS converges to the optimal solutions with much less time compared to BFS.

2) Partition Parameter: We conduct ablation studies of our method in the Manhattan City network to study the effects of the number of regions, the minimum size of each region and the partition interval. The number of regions and the minimum size of each region directly affect the partition of the road network. Meanwhile, the minimum size of each region also affects the number of regions, i.e., a smaller size leads to more regions. If the minimum size of each region is 1, AFMA2C is simplified to *independent* hierarchical control. In other words, fewer agents in each region will result in less intra-regional cooperation. When the minimum size of each region is the whole network size, AFMA2C is equivalent to *centralized* hierarchical control, which is inefficient. Indeed, an intermediate value between them will achieve the best performance. As shown in Figure 12(a-b), when the number of regions is 4 and the minimum size of each region is 12, AFMA2C obtained better performance.

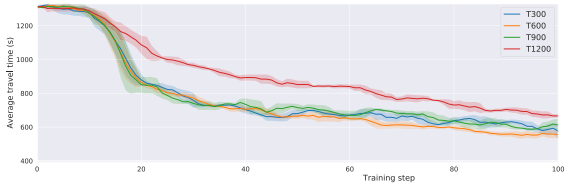
The partition interval is the interval time of performing network partition. As shown in Figure 12(c), when the interval



(a) The number of regions



(b) The minimum size of each region



(c) The partition interval

Fig. 12. Ablation results for partition hyper-parameters.

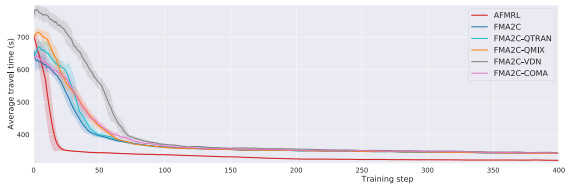
time is too large, AFM2C will degenerate into the static network partition and the performance drops sharply. To obtain the optimal interval time, prior knowledge about the usual duration of the pattern of traffic flow may be needed.

3) Manager-Level Coordination: It is also worth studying the performance of the coordination mechanism in the manager-level. In our paper, we believe the best case is when the intra-regional connection is high, the cooperation of workers in the same region will become closely. This is because managers do not directly control the traffic lights. Thus, if the intra-regional connection is sparse (i.e., the roads between them are not congested), the coordination of workers in the same region is less effective. As shown in Figure 13(a-b), adaptive network partition is necessary to the traffic signal control, as shown in the comparison of AFMA2C and other methods. Therefore, it is necessary to adaptively partition the network to make the intra-regional connection tight.

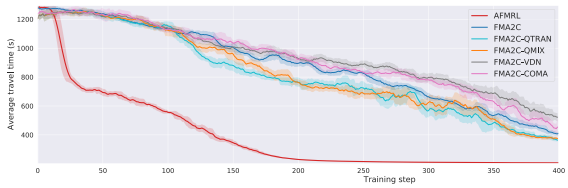
In addition, the multi-agent learning technique is also beneficial. To confirm this, we conduct ablation experiments that consider other multi-agent learning techniques at the manager level, i.e., VDN, COMA, QMIX, QTRAN. Note that the FMA2C employ the MA2C for the manager-level coordition. As shown in Figure 13(a-b), in a complex and larger road network (Manhattan City), effective inter-regional cooperation is also beneficial, as shown in the comparison of FMA2C-QTRAN and FMA2C-VDN. All in all, although inter-regional cooperation (the multi-agent technique) is useful, adaptive network partition is more critical for the overall performance.

F. Discussions on Real-Time Performance

As mentioned, TSC is a time-sensitive domain and the methods for TSC should work in real-time. Here, our method



(a) Hangzhou City



(b) Manhattan City

Fig. 13. Ablation results for manager-level coordination.

takes 8.9s to compute the new partitions and only 0.16s to compute the control actions in the 16×3 city network. In the real-world, the pattern of traffic flow usually continues for a period of time and does not change quickly (e.g., morning-evening rush hours). Hence we do not need to perform network partition very frequently, i.e., every 300 timesteps in our experiments. The interval of traffic signal switching generally takes tens of seconds for traffic to respond. In addition, the communication delay of collecting the network-level real traffic information is very small with the current transmission technology. Therefore, time is sufficient for our method to compute a new partition, and time for choosing a control action (i.e., time for neural network inference) is negligible ($<$ 1s). Moreover, MCTS is indeed an anytime search algorithm and can produce better results if more time is given. Therefore, our method can meet the real-time demand for TSC.

VI. Conclusion

In this article, we proposed a feudal MARL with adaptive network partition for TSC. Firstly, we introduced two approaches based on GNN and MCTS respectively, to partition the traffic network into several regions, fitting for dynamic traffic flow. Next, we use the GNN to predict the criteria of choosing the best partition in the building process of MCTS. With it, we select a good network partition in a feasible time compared to the enumeration method. Then, we combine Qmix with GNN to extract the features of variant dimensions region and cooperative control the entire traffic network. Finally, we apply the feudal MARL to each region for global cooperation control. In the experiment, we benchmark in a synthetic and three real-world traffic networks. The results show better performance of our approach over several state-of-the-art TSC methods and our advantage in the large-scale traffic network. In the future, we plan to further improve the partition methods with prior knowledge and test it on real-world applications.

References

[1] P. Koonce and L. Rodegerdts, "Traffic signal timing manual," Federal Highway Admin., Washington, DC, USA, Tech. Rep. FHWA-HOP-08-024, 2008.

[2] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in *Proc. Learn., Inference Control Multi-Agent Syst. (NIPS)*, 2016, pp. 1–9.

[3] H. Wei et al., "CoLight: Learning network-level cooperation for traffic signal control," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1913–1922.

[4] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.

[5] L. A. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 412–421, Jun. 2011.

[6] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2496–2505.

[7] H. Wei et al., "PressLight: Learning max pressure control to coordinate traffic signals in arterial network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1290–1298.

[8] C. Chen et al., "Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 3414–3421.

[9] J. Ma and F. Wu, "Feudal multi-agent deep reinforcement learning for traffic signal control," in *Proc. 19th Int. Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2020, pp. 816–824.

[10] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proc. Eur. Conf. Mach. Learn.*, 2006, pp. 282–293.

[11] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[12] C. Gershenson, "Self-organizing traffic lights," 2004, *arXiv:nlin/0411066*.

[13] P. Varaiya, "The max-pressure controller for arbitrary networks of signalized intersections," in *Proc. Adv. Dyn. Netw. Model. Complex Transp. Syst.*, 2013, pp. 27–66.

[14] G. Zheng et al., "Learning phase competition for traffic signal control," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1963–1972.

[15] H. M. A. Aziz, F. Zhu, and S. V. Ukkusuri, "Learning-based traffic signal control algorithms with neighborhood information sharing: An application for sustainable mobility," *J. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 40–52, Jan. 2018.

[16] B. Xu, Y. Wang, Z. Wang, H. Jia, and Z. Lu, "Hierarchically and cooperatively learning traffic signal control," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 1, 2021, pp. 669–677.

[17] M. A. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 1151–1158.

[18] D. Liu, S. Baldi, W. Yu, and G. Chen, "On distributed implementation of switch-based adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 7218–7224, Jul. 2022.

[19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[20] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1–9.

[22] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.

[23] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4805–4815.

[24] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*.

[25] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1106–1114.

[26] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2357–2366.

[27] W. Yu et al., "Learning deep network representations with adversarially regularized autoencoders," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2663–2671.
[28] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*.
[29] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.
[30] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[31] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Exp. Syst. Appl.*, vol. 207, Nov. 2022, Art. no. 117921.
[32] J. Ye, J. Zhao, K. Ye, and C. Xu, "How to build a graph-based deep learning architecture in traffic domain: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 3904–3924, May 2022.
[33] Q. Zhang, Q. Jin, J. Chang, S. Xiang, and C. Pan, "Kernel-weighted graph convolutional network: A deep learning approach for traffic forecasting," in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 1018–1023.
[34] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial–temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 922–929.
[35] L. Ge, H. Li, J. Liu, and A. Zhou, "Temporal graph convolutional networks for traffic speed prediction considering external factors," in *Proc. 20th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2019, pp. 234–242.
[36] B. Yu, M. Li, J. Zhang, and Z. Zhu, "3D graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting," 2019, *arXiv:1903.00919*.
[37] L. Yan et al., "Employing opportunistic charging for electric taxicabs to reduce idle time," in *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 2, no. 1, Mar. 2018, pp. 1–25.
[38] X. Geng, X. Wu, L. Zhang, Q. Yang, Y. Liu, and J. Ye, "Multi-modal graph interaction for multi-graph convolution network in urban spatiotemporal forecasting," 2019, *arXiv:1905.11395*.
[39] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, "STG2Seq: Spatial–temporal graph to sequence model for multi-step passenger demand forecasting," 2019, *arXiv:1905.10069*.
[40] T. Nishi, K. Otaki, K. Hayakawa, and T. Yoshimura, "Traffic signal control based on reinforcement learning with graph convolutional neural nets," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 877–883.
[41] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong, "STMARL: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2228–2242, Jun. 2022.
[42] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.
[43] Z. Zhao, H. Fang, Z. Jin, and Q. Qiu, "GISNet: Graph-based information sharing network for vehicle trajectory prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–7.
[44] H. Wei, G. Zheng, V. Gayah, and Z. Li, "A survey on traffic signal control methods," 2019, *arXiv:1904.08117*.
[45] F. Wu and S. D. Ramchurn, "Monte-Carlo tree search for scalable coalition formation," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 407–413.
[46] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.
[47] T. Chu, S. Qu, and J. Wang, "Large-scale traffic grid signal control with regional reinforcement learning," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 815–820.
[48] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[49] H. Zhang et al., "CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario," in *Proc. WWW*, 2019, pp. 1–5.
[50] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in Applied Self-Organizing Systems* (Advanced Information and Knowledge Processing). London, U.K.: Springer, 2008, pp. 41–50.
[51] S. Jiang, Y. Huang, M. Jafari, and M. Jalayer, "A distributed multi-agent reinforcement learning with graph decomposition approach for large-scale adaptive traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14689–14701, Sep. 2022.
[52] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, *Recent Advances in Graph Partitioning*. Cham, Switzerland: Springer, 2016.
[53] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 291–307, Feb. 1970.
[54] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," in *Proc. 36th Annu. ACM/IEEE Design Autom. Conf.*, Aug. 1999, pp. 343–348.

Jinming Ma received the bachelor's degree from Tianjin University in 2018. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include reinforcement learning and multi-agent reinforcement learning.

Feng Wu received the B.E. and Ph.D. degrees from the University of Science and Technology of China (USTC), in 2006 and 2011, respectively. He is an Associate Professor with the School of Computer Science and Technology, USTC. His research interests include planning under uncertainty, multi-agent systems, reinforcement learning, and robotics. He has published over 60 refereed papers on these topics in prestigious AI journals and conferences.