Spectrum Allocation and Scheduling in Conflict Graphs via Scalable Multi-agent Reinforcement Learning

Yiming Zhang, Dongning Guo

***Abstract*—This paper introduces a novel approach to scheduling in conflict graphs by leveraging a fully scalable multi-agent reinforcement learning (MARL) framework. The objective is to minimize average delays in the presence of stochastic packet arrivals to links, where scheduling conflicts arise if two vertices (or links) are assigned to the same spectrum sub-band in the same time slot. The problem is formulated as a decentralized partially observable Markov decision process (Dec-POMDP), with the implementation of the multi-agent proximal policy optimization (MAPPO) algorithm to optimize scheduling decisions. To enhance performance, advanced recurrent structures in the neural network are integrated. The proposed MARL solution allows for both decentralized training and execution, facilitating scalability to large networks. Extensive simulations conducted across diverse conflict graphs demonstrate the superior performance of the approach in terms of throughput and delay compared to established schedulers under varied traffic conditions.**

***Index Terms*—Markov decision process; multi-agent reinforcement learning (MARL); recurrent neural networks; stochastic traffic; wireless networks.**

I. Introduction

This paper addresses the challenge of scheduling in conflict graphs, a crucial problem with broad applications across various domains [1], [2]. This problem involves allocating resources like time slots and/or radio spectrum sub-bands to conflicting tasks or events while considering both mutual interference and operational constraints. In wireless communication networks, conflict graphs serves as an effective abstraction to represent interference and constraints between different links. Our goal is to develop a distributed, efficient, and scalable method for scheduling wireless links that interfere with each other across multiple frequency sub-bands.

Traditional method like the MaxWeight algorithm achieves optimal throughput in a conflict graph [3], but requires identifying all maximum independent sets within the graph, which is an NP-complete problem [4]. Low-complexity heuristic methods such as longest-queue-first (LQF) provide simpler alternatives but often support only a portion of the capacity region. A recent study [2] adopted graph neural networks, leveraging topological information to address similar scheduling problems in conflict graphs, showcasing the potential of machine learning in enhancing scheduling strategies. However, all these methods are centralized, necessitating the collection of information from all links in the graph and hence impractical in large communication. A low-complexity distributed scheduling method called queue-length-based carrier-sense multiple access (Q-CSMA) [5] was proposed as a practical solution. We use it as one of the benchmarks in our simulation.

Aforementioned works and recent machine learning-based scheduling studies [2], [6]–[8] have primarily focused on throughput maximization and use sum-rate as the key performance metric. In this work, we prioritizes the average packet delay as the quality of service (QoS) metric for two main reasons. Firstly, wireless networks often operate under lighter traffic conditions than their maximum throughput capacity allows, making latency a more relevant measure of user experience. Secondly, high throughput does not necessarily eliminate significant packet delays, which can occur due to unbalanced scheduling that disproportionately favors certain links.

Formulating a tractable delay minimization problem is a longstanding challenge. This paper adopts a data-driven, model-free approach and leverage reinforcement learning (RL), which may offer substantial benefits over traditional optimization approaches. First, it circumvents the difficulty of directly formulating the delay minimization problem and the corresponding computational complexity in potentially high-dimensional, non-convex optimization. Second, RL-based scheduling leverages historical data and interactions, thereby offering a richer understanding of system dynamics over time, unlike traditional methods which focus on optimizing objectives based on the current state alone. Third, this historical perspective also enables consideration of long-term utility in decision-making processes. While RL has been successfully employed in various network resource allocation problems (see, e.g.,) [9]–[11], these RL-based solutions typically converge to a fixed optimal allocation such as the power levels of transmitters [9], [11] or control variables in unmanned aerial vehicles (UAVs) [10]. In contrast, our work aims to learn flexible and adaptable policies that can map dynamic traffics to a broad spectrum of actions, thereby avoiding the limitations of static solutions and continually adapting to fluctuating traffic loads and varying channel conditions.

Multi-agent reinforcement learning (MARL) expands the scope of RL by introducing multiple agents that interact and learn simultaneously to achieve desirable rewards. While [12] studies the case where each agent controls one link as in an ad hoc network, this paper develops an MARL framework

The authors are with Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208, USA (e-mail: {yimingzhang2026, dguo}@northwestern.edu).

The work was supported in part by the National Science Foundation (NSF) under grant Nos. 2003098 and 2216970, a gift from Intel Corporation, and also the SpectrumX Center under NSF grant No. 2132700.

for cellular wireless networks, where each agent manages all links in a cell. These agents needs to learn effective scheduling policies considering the intra-cell and inter-cell interference constraints described by the conflict graph. The inherent need for agents to coordinate their actions and make jointly good decisions makes MARL a natural fit for this setting.

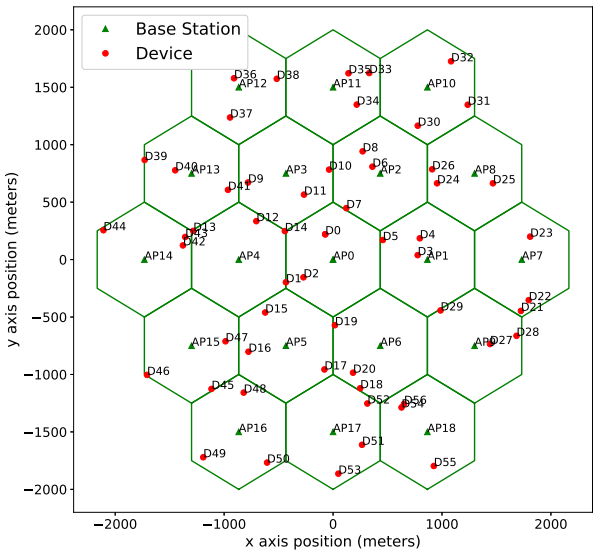
One key challenge of MARL is the scalability issue caused by the exponentially large joint-action space. Function approximation techniques such as linear methods [13] and neural networks [14] have been used to manage the expansive Q-table in temporal difference learning.

In prior work [12], we utilized centralized training and distributed execution, where the size of a critic neural network is proportional to the network size. In this paper, we adapt the previous framework, using only neighborhood information during both training and execution phases. The adjustment keeps the training costs and neural network size constant for each agent, thus ensuring scalability.

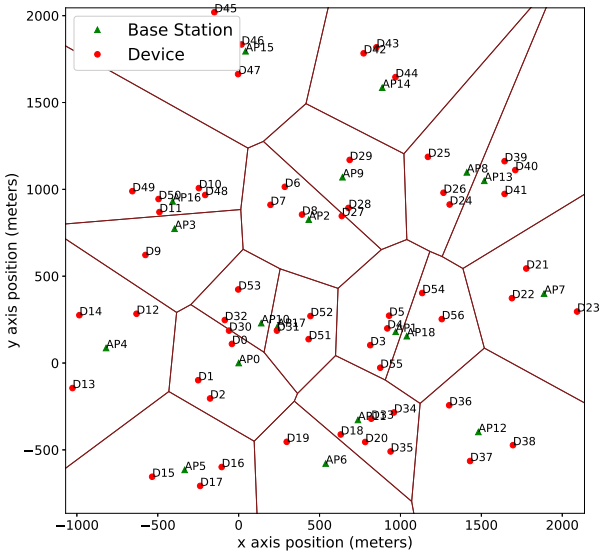
Another challenge arises from the limited information exchange in a practical network. This limitation, often due to constraints on communication overhead or excessive delays over multiple hops, means that each agent may only have access to observations made in its neighborhood. Consequently, agents must learn local policies that map their limited observations to local actions, necessitating a distributed approach. Recently, MARL's application in networked systems, with agents relying on local observations, has been a major research focus [7], [15], [16]. Unlike in [16], which makes an unrealistic assumption that one agent's queue length is independent of other agents' actions, here we allow arbitrary dependence on neighboring agents' actions.

To address scalability and the need for decentralization, we approach MARL in conflict graphs as distributed learning in a decentralized partially observable Markov decision process (Dec-POMDP) framework. We employ a widely adopted on-policy algorithm multi-agent proximal policy optimization (MAPPO) and integrate recurrent structures in the neural network. These recurrent structures enable agents to better estimate the underlying state by encoding historical transitions.

The remainder of this paper is organized as follows. We describe the system model and formulate the learning problem in Section II. In Section III, we introduce an RL framework. In Section IV, we discuss the simulation setup and numerical results. Concluding remarks are given in Section V.



(a)



(b)

Fig. 1: Two networks with 19 cells serving 57 devices. (a) A regular hexagonal deployment where each AP serves 3 devices; (b) A random deployment.

II. System Model and Problem Formulation

A. System model

We consider the downlink broadcast channel within a wireless cellular network comprising N devices distributed over K cells. Let $\mathcal{K} = \{1, 2, \ldots, K\}$ and $\mathcal{N} = \{1, 2, \ldots, N\}$ denote the set of cell indices and device indices, respectively. Each cell hosts an AP $k \in \mathcal{K}$. Devices are allocated to their nearest AP for service, with the serving AP of device n is denoted as $b_n \in \mathcal{K}$. Thus, $\mathcal{N}_k = \{n \in \mathcal{N} \mid b_n = k\}$ denotes all the devices served by AP k. Let time be slotted, an AP k transmits data packets to devices in $\mathcal{N}_k$ over a set of consecutive discrete time slots with slot duration T. We assume that that all transmitters and receivers are equipped with a single antenna.

Examples of cellular networks are given in Fig. 1. Fig. 1a depicts a regular network with 57 devices deployed across 19 hexagonal cells, where each access point (AP) serves three devices within its cell. Fig. 1b illustrates a network where both APs and devices are randomly deployed, in which the number of devices served by an AP ranges from 1 to 5. In general, a

network with N devices has N AP-to-device links.

B. Conflict Graph

Wireless links may have conflict with one another if activated simultaneously on the same sub-band due to the wireless interference or physical constraints. To model this, consider a directed conflict graph denoted as $\mathcal{G} = (\mathcal{I}, \mathcal{E})$, where each vertex in $\mathcal{I}$ represents a link, and an edge $(i, j) \in \mathcal{E}$ with $i, j \in \mathcal{I}$ and $i \neq j$, indicates link i would cause conflict to link j if they are activated on the same sub-band.

As an example, Fig. 2a depicts a symmetric network deployment where each AP serves 2 devices. Fig. 2b depicts a conflict graph as an abstraction of the deployment, in which D1,..., D8 represent 8 links. Since APs can not serve multiple links on the same sub-band simultaneously, conflicts are bidirectional within the same cell, highlighting intra-cell conflict. For devices positioned at the cell boundary, the corresponding links would be strongly interfered by the activation of nearby AP. For examples, the activation of AP2, would cause interference to D2, therefore, the links served by AP2, i.e. D3 and D4, would cause conflict to D2, illustrating the inter-cell conflict.

C. Cell-based Agent

In this work we study the performance of cell-based agents, each associated with an AP and also denoted by $k \in \mathcal{K}$. These agents are responsible for scheduling transmissions for all devices within their respective cells. For instance, as shown in Fig. 2b, agent 1 needs to schedule transmissions for D1 and D2. A neighborhood is defined for each agent, which includes the agent itself and some other agents, referred to as its neighbors. Here we simply let agent k's neighborhood be defined to include all agents whose links conflict with those served by agent k. For instance, in Fig. 2b, agent 1's neighborhood includes agents 2 and 4, while agent 2's neighborhood includes agents 1 and 3, and so on. Let l_k denote the number of neighbors agent k has, and let $\nu_{k,1}, \ldots, \nu_{k,l_k}$ denote their indexes. Let $\mathcal{C}(k) = \{k, \nu_{k,1}, \ldots, \nu_{k,l_k}\}$ denote agent k's neighborhood, which always includes the agent itself. Agent k utilizes information from $\mathcal{C}(k)$ to schedule transmissions for all devices in its service area.

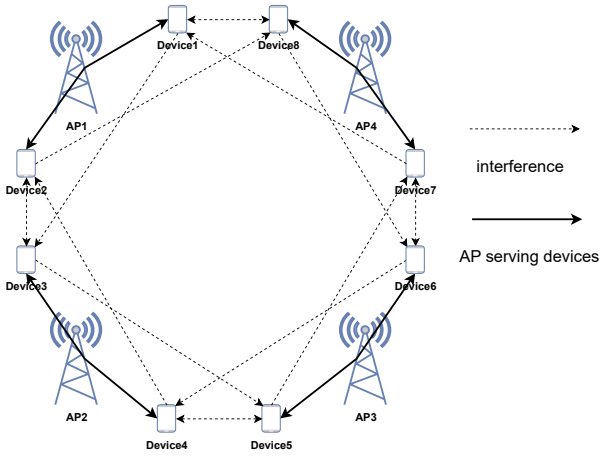
Each link operates a first-in-first-out (FIFO) queue. Let $X_n^{(t)}$ denote the number of newly arrived packets to link n at the beginning of time slot t. We assume that $X_n^{(t)}$ is independent across all links n and time slot t, and all packets are of identical size. We adopt the usual collision model, where a packet transmission on a specific sub-band by a link during a time slot succeeds if and only if no other conflicting link transmits on the same sub-band in that slot. Without loss of generality, we assume that all links have unit capacity on each sub-band, i.e., a scheduled link can transmit up to one packet using one sub-band in one time slot.

Let H represent the total number of sub-bands, with $h \in \{1, \ldots, H\}$ specifying a particular sub-band. We assume all the sub-bands are orthogonal. In each time slot t, the agent k is tasked with making scheduling decision $a_k^{(t)}$ across all sub-bands simultaneously. $a_k^{(t)} = \{a_{k,1}^{(t)}, \ldots, a_{k,H}^{(t)}\}$. For sub-band h, the scheduling decision $a_{k,h}^{(t)}$ would be selected from

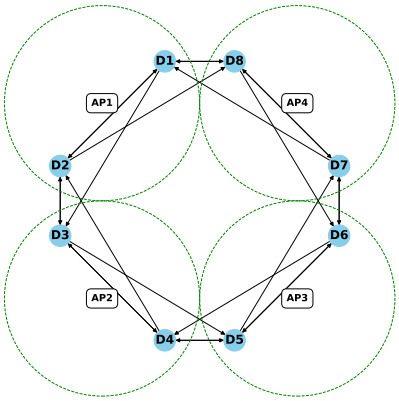
$$\{0, 1, , \ldots, |\mathcal{N}_k|\}. \tag{1}$$

where a decision of 0 indicates that no links are activated during time slot t, or alternatively, an agent may select one of the links within its cell for transmission on sub-band h. Crucially, global-to-local index mapping strategy is employed in implementation, converting decisions from the local index to the global index of the link. Specifically, we define a bijective mapping f maps each global index to a corresponding AP's index together with the local index in that cell, formally defined as $f : \{1, \ldots, N\} \rightarrow \{(1,1), \ldots, (|\mathcal{N}_k|, 1), (1,2) \ldots, (|\mathcal{N}_k|, K)\}$. For example, in Fig. 2b, D5 and D6 are the two links in cell 3, so $f(5) = (1,3)$, $f(6) = (2,3)$, and $f^{-1}(1,3) = 5$.

Let $\mu_{n,h}^{(t)}$ and $m_{n,h}^{(t)}$ represent the scheduling decision and the number of successfully transmitted packets of link n on h-th sub-band in time slot t, respectively. The binary variable $\mu_{n,h}^{(t)} = 1$ indicates the link n is scheduled for transmission on sub-band h in time slot t. This condition holds true when



(a)



(b)

Fig. 2: Examples of network deployment and abstracted conflict graph. (a) 4 APs in symmetry deployment and each serving 2 devices; (b) The abstracted conflict graph.

$f^{-1}(a^{(t)}_{b_n,h}, b_n) = n$. Consequently, the number of packets successfully transmitted, $m^{(t)}_{n,h}$, is determined as follows:

$$m^{(t)}_{n,h} = \begin{cases} 1, & \text{if } \mu^{(t)}_{n,h} = 1, \mu^{(t)}_{i,h} \neq 1 \text{ for all } (i,n) \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Specifically, $m^{(t)}_{n,h} = 1$ indicates that link n is scheduled for conflict-free transmission on the h-th sub-band at time slot t. If there is a conflict or the link is not scheduled for transmission, then $m^{(t)}_{n,h} = 0$. Given these assumption, we can accurately represent queueing dynamic for each link. The queue length of link n at the end of slot t is expressed as follows:

$$q^{(t)}_n = \max\left(0,\, q^{(t-1)}_n + X^{(t)}_n - \sum_{h=1}^{S} m^{(t)}_{n,h}\right). \tag{3}$$

This representation signifies the queue length of agent n at the moment of measuring the local observation. We assume that $q^{(0)}_n = 0$ since the queues start empty.

D. Problem Formulation and QoS

Our objective is to minimize the packet delay of aforementioned system. We formulate the problem as distributed learning in Dec-POMDP. Consider a network of K agents and N links. Let the global state space be denoted as $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_k$. Let $s^{(t)} = (s^{(t)}_1, \ldots, s^{(t)}_K) \in \mathcal{S}$ denote the global state at a given point t. The state entry s_k comprises queue information for all the links managed by AP k's. Specifically, define the queue information ζ_n of link n as:

$$\zeta^{(t)}_n = q^{(t-1)}_n + X^{(t)}_n. \tag{4}$$

Then

$$s^{(t)}_k = \left\{\zeta^{(t)}_{f^{-1}(1,k)}, \cdots, \zeta^{(t)}_{f^{-1}(|\mathcal{N}_k|,k)}\right\}. \tag{5}$$

The local observation of agent k, denoted as $O_k(s) = (s_k, s_{\nu_{k,1}}, \ldots, s_{\nu_{k,l_k}})$, consists of entries of the global state. Let $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_K$ represent the joint action space of all agents. In each transition, agent k selects an action $a_k \in \mathcal{A}_k$ based on its local observation. Let $P(s' \mid s, a)$ represent the transition probability from state s to s' given the joint action $a = (a_1, \ldots, a_K)$. The reward function of agent k, denoted as $R_k(s, a, s')$, depends on the state, transition, and the joint action. We express the total discounted reward of a given trajectory of the state and action from time slot t onward as:

$$\sum_{\tau=0}^{\infty} \gamma^\tau R^{(t+\tau)}_k \left(s^{(t+\tau)}, a^{(t+\tau)}, s^{(t+1+\tau)}\right). \tag{6}$$

In slot t, the control policy of agent k, denoted as π_{k,θ_k}, which is a neural network parameterized by weights θ_k, maps the local observation

$$O^{(t)}_k(s) = (s^{(t)}_k, s^{(t)}_{\nu_{k,1}}, \ldots, s^{(t)}_{\nu_{k,l_k}}). \tag{7}$$

to the action $a^{(t)}_k$. For agent $k \in \{1, \ldots, K\}$ and sub-band $h \in \{1, \ldots, H\}$, the control policy decision variable $a^{(t)}_{k,h}$ selected from (1) indicates the agent k's decision on sub-band h in time slot t. The policy π_{k,θ_k} is trained using a collection of state, action, and reward trajectories. In this work, all agents are trained in parallel. The goal is that once trained, the agents have learned policies that lead to highly desirable total reward.

In this work, we define learning objectives using queue lengths as surrogates for the packet delay. Evidently, the longer a link' queue length is, the longer the packet delays are. For a given traffic, the delay QoS is a more relevant metric than the total throughput. Specifically, the direct contribution of agent k to the queue length objective can be expressed as:

$$u^{(t)}_k(s^{(t-1)}, a^{(t)}, s^{(t)}) = -\sum_{i \in \mathcal{N}_k} q^{(t)}_i. \tag{8}$$

We also include the utilities of agent k's neighbors as indirect contributions in order to encourage joint decisions that lead to mutually beneficial outcomes. The agent k's reward function is expressed as:

$$R^{(t)}_k \left(s^{(t)}, a^{(t)}, s^{(t+1)}\right) = \sum_{i \in \mathcal{C}(k)} u^{(t)}_i. \tag{9}$$

Generally, the reward of agent k, denoted as R_k, is a function of global states, transitions and actions. In our design, it can be computed locally using only queue lengths from agent k and its neighbors. For example, the reward for agent 1 in Fig. 2b in slot t is equal to $-\left(q^{(t)}_1 + q^{(t)}_2 + q^{(t)}_3 + q^{(t)}_4 + q^{(t)}_7 + q^{(t)}_8\right)$, which agent 1 can compute using its own information and information from neighboring agents 2 and 4.

III. A Reinforcement Learning Framework

In this paper, we adopt a multi-agent reinforcement learning setting where agents interact with an environment in discrete time steps to learn a policy. We employ an on-policy algorithm, MAPPO, which has demonstrated success in solving various cooperative multi-agent tasks [17].

We consider two different training methods. In the first method, agents individually train distinct actor and critic networks using MAPPO based on their own interaction trajectories, facilitating a fully decentralized policy tailored to each agent's specific neighborhood and traffic conditions. In the second method, we train one common neural network as a shared policy for all agents. While this shared policy typically achieves a lower average reward than the individualized method, its total computational complexity across all agents is substantially lower. This shared approach is particularly effective in scenarios where agents are homogeneous and encounter similar local topology and traffic conditions. In the following, we explore the intricacies of neural networks, recurrent structures and decentralized training crucial to our MARL method.

A. Actor Network

The actor network, parameterized by θ, represents the policy π_θ for each agent. It maps the local observation O_k to a categorical distribution over discrete actions in our specific setting. Let θ_{old} denote the parameter values of the policy network from the previous iteration. Let B represent the

training batch size. For agent k and time slot t in a batch, we define

$$r_{\theta,k}^{(t)} = \pi_\theta \left(a_k^{(t)} \mid O_k^{(t)}\right) \Big/ \pi_{\theta_{\text{old}}} \left(a_k^{(t)} \mid O_k^{(t)}\right). \tag{10}$$

Let $\mathcal{H}(\cdot)$ denote the Shannon entropy of a probability mass function. We use the generalized advantage estimation (GAE) method in [18, Eq. 15] to compute the advantage function, denoted as $A_k^{(t)}$. The training objective in each update for the actor is to maximize the following function:

$$\begin{aligned} J(\theta) =& \frac{1}{BK} \sum_{t=1}^{B} \sum_{k=1}^{K} \min \left(r_{\theta,k}^{(t)} A_k^{(t)}, c_\epsilon \left(r_{\theta,k}^{(t)}, 1\right) A_k^{(t)}\right) \\ &+ \sigma \frac{1}{BK} \sum_{t=1}^{B} \sum_{k=1}^{K} \mathcal{H}\left(\pi_\theta \left(\cdot \mid O_k^{(t)}\right)\right) \end{aligned} \tag{11}$$

where σ denotes a hyperparameter and

$$c_\epsilon(x, y) = \min(\max(x, y - \epsilon), y + \epsilon) \tag{12}$$

represents a clipping function.

B. Critic Network

The critic network in MAPPO, parameterized by ϕ, maps a local state O_k to $V_\phi(O_k)$, which is an estimate of the expected return from a given state of agent k. The training objective in each update for the critic network is to minimize the loss function:

$$\begin{aligned} L(\phi) = \frac{1}{BK} \sum_{t=1}^{B} \sum_{k=1}^{K} \max \Bigg[& \left(V_\phi \left(O_k^{(t)}\right) - \hat{R}^{(t)}\right)^2, \\ & \left(c_\epsilon \left(V_\phi \left(O_k^{(t)}\right), V_{\phi_{\text{old}}} \left(O_k^{(t)}\right)\right) - \hat{R}^{(t)}\right)^2 \Bigg] \end{aligned} \tag{13}$$

where $\hat{R}^{(t)}$ represents the discounted reward-to-go, which is defined as the cumulative sum of discounted rewards from a specific time slot t until the end of an episode.

C. Recurrent Structure

In our Dec-POMDP formulation, each agent's input is limited to local observations within its neighborhood. In order to adapt to environmental dynamics beyond a single local observation, we consider utilizing recurrent structures like long short-term memory (LSTM) or gated recurrent unit (GRU) in our neural networks. These structures help agents to infer the impact of other agents' actions and make decisions utilizing historical states and actions. The inclusion of sequential memory empowers the agent's ability to incorporate long-term consequences of its actions and leveraging the accumulated knowledge stored within the memory.

In our implementation, both the actor and critic networks are recurrent neural networks (RNNs). We aggregate the loss function defined in (13) over time and use the RNN states as input to these networks. Back-propagation through time (BPTT) is employed for training and a detailed pseudocode illustrating the training phase is found in [17]. During the training phase, all actor and critic networks are updated for a fixed number of steps for each training episode.

D. Decentralized Training and Distributed Execution

The paradigm of centralized training and distributed execution is widely adopted to addresses challenges of non-stationarity and scalability. Under this paradigm, global information is used in the training stage to enhance the learning phase, whereas agents rely solely on local information during execution. The conventional MAPPO follows this paradigm, where the computation of the value function $V_\phi(s)$ during training uses the global state s as input to the critic network. However, it is impractical for a central controller to gather timely global information in many real-world scenarios.

Inspired by the scalable framework suggested in [15], we modify centralized training by feeding local observations into the critic network in lieu of the global state. Specifically, for each agent k, the critic network ϕ_k is fed with its local observation O_k, which contains the information of itself and its neighbors. We assume that the neighborhood size is bounded by a constant, ensuring that the input dimension to the neural network is fixed. This adjustment ensures full scalability as the training cost per agent remains constant and does not escalate with an increase in the number of agents. The fact the cost is constant is due to two primary reasons: the fixed size of the policy and critic networks, as they process fixed-dimension inputs and outputs, and the stable number of interactions within each agent's neighborhood, preventing any increase in computational complexity as the agent count grows. Our experiments validate the effectiveness of this approach.

E. Characteristics of States, Actions and Rewards

This section highlights the salient features of the state space, action space, and reward function to enhance clarity and to ensure reproducibility of our work, despite having previously discussed and defined them. To ease implementation, we introduce dummy links to maintain identical state and action space dimensions for all agents, regardless of the number of devices they serve, under the practical assumption this number is capped by a constant.

State Space: Following the Dec-POMDP framework, the global state in time slot t can be represented as $s^{(t)} = \left(s_1^{(t)}, \ldots, s_K^{(t)}\right)$, where $s_k^{(t)}$ is defined in (5). Given the constraints of our Dec-POMDP setting, an agent k only has access to information about itself and its neighbors. Therefore, the local observation is expressed in (7). Clipping is employed here to make the observation to be a coarsely quantized function of the queue length, which guarantee the state space to be finite.

Action Space: The action space dimension for each agent is determined by the number of devices $|\mathcal{N}_k|$ and the number of sub-bands H. The action decision of agent k allocate each sub-band to one of the devices as shown in (1)(where a value of 0 indicates that no devices are scheduled). Consequently, there are $H^{|\mathcal{N}_k|+1}$ possible actions available to each agent. If H is not too small, the large action space renders tabular reinforcement learning methods impractical [16]. The agent's decision-making allocates sub-bands to links, considering constraints from neighboring agents' actions to optimize transmission performance.

Reward Function: In our traffic-driven approach, the objective is to minimize the average packet delay, which is equivalent to minimizing the long-term queue length. The direct contribution of agent k to the queue length objective has been defined in (8) and the agent k's reward function is a function of queue length of itself and its neighbors at time slot t defined in (9).

State transition: The next global state $s^{(t)}$ is determined by the joint action $a^{(t)}$ and the randomness of system. More specifically, each entry is a function of queue length and the transition is expressed in (3).

IV. Simulation Results and Analysis

A. Simulation Setup

We model the traffic arriving to link n as a discrete-time memoryless Poisson process. Specifically, let the number of packet arrivals to agent n in time slot t, denoted by $X_n^{(t)}$, be an independent Poisson random variable with mean λ_n. The experiments are conducted on three distinct conflict graphs :

1) The conflict graph depicted in Fig. 2b, where 8 devices and 4 APs deployed symmetrically.
2) A conflict graph as an abstraction of a cellular network depicted in Fig. 1a. Here each user device has a direct link to its nearest AP. We define the downlink channel gain from AP b_n to device n as $g_{b_n \to n}$. If the gain difference $g_{b_n \to n} - g_{k \to n}$ falls below a certain threshold, then link n is considered to be in conflict with links in the set $\mathcal{N}_k$.
3) A conflict graph as an abstraction of a cellular network depicted in Fig. 1b. Compared with Fig. 1a, the devices and APs are deployed randomly. Each AP serves a variable number of devices, reflecting a less structured network topology.

The performance of the proposed MARL-based scheduler is evaluated and compared in terms of QoS against two benchmarks. The first benchmark scheme, referred to as local longest queue (LLQ), schedules a link for transmission if it has a longer queue than all links that it has a conflict with; in case of a tie between multiple links, each link is scheduled with a uniform probability independently among these links. The second benchmark scheme is Q-CSMA [5], where links perform carrier sensing prior to transmission, ensuring that all scheduled links form an independent set, and then each enabled link transmits with a certain probability, where the probability is updated based on its queue length.

Throughout this section, we assume the spectrum is divided into $H = 3$ subbands. we give both benchmark schemes the additional advantage of having H rounds of information sharing before actual transmissions in each time slot t. This is equivalent to letting both schemes sequentially determine the actions on a subband depending on the transmission outcome of previous subbands. In contrast, our MARL method allows agents to share information with their neighbors only once before deciding on their transmissions on all sub-bands simultaneously. As we shall see, the MARL method still outperforms than the benchmarks.

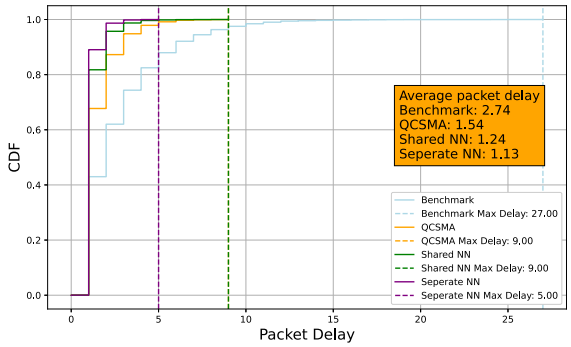


Fig. 3: CDFs of packets delay in 8-link conflict graph under light traffic.

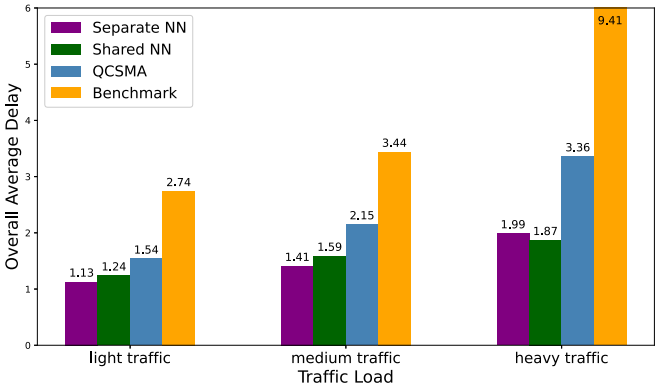


Fig. 4: Average packet delay of 8-link conflict graph.

B. Training and Testing

We consider a centrally trained policy as well as individual policies trained separately. We set the duration of a training episode to 2,000 time slots. Whenever the queue length of any link surpasses a predefined threshold, it signifies a congested or very unfavorable traffic situation. In such instances, the current episode is curtailed, and a new episode is started. The purpose here is to avoid being trapped in adverse queueing conditions before the scheduler is trained well.

During testing or deployment, we set the duration of a testing episode to 5,000 time slots and adopt the average packet delay as the performance metric (when the queue is perceived to be stable).

C. Performance analysis

We compare the average packet delays attained by the MARL method and the LLQ and Q-CSMA benchmark. The packet delays for each method were recorded and the cumulative distribution functions (CDFs) of these packet delays are plotted in Figure 3, where all methods were tested under a light traffic condition. The CDFs of both MARL approaches—utilizing either a shared or separate policies—dominate those of the benchmarks.

Notably, more than 80% of the packets are transmitted immediately (within just one time slot) using either of the

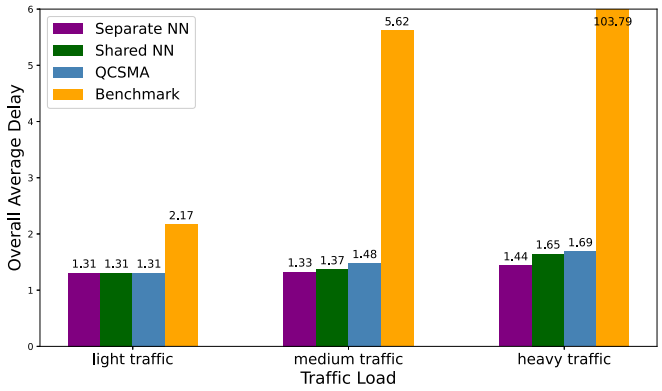


Fig. 5: Average packet delay of 57-link conflict graph abstracted from Fig. 1a.

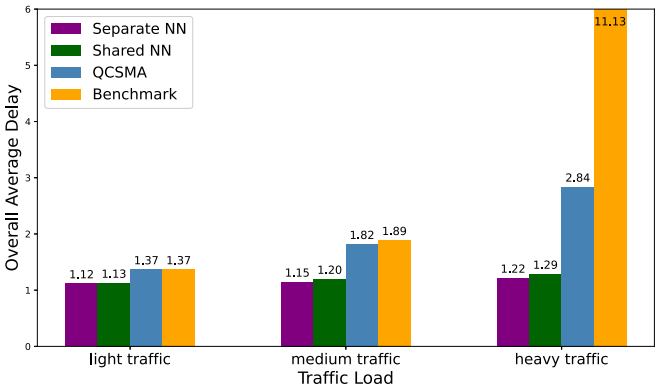


Fig. 6: Average packet delay of 57-link conflict graph abstracted from Fig. 1b.

MARL methods. The average packet delay of the MARL method with a shared policy is 1.24, which is 19.5% less than that of Q-CSMA (1.54) and 54.7% than that of LLQ (2.74). If agents use separately trained policies, the average delay is even smaller at 1.13. Additionally, the maximum packet delay for MARL with separate policies is significantly lower, at 5 time slots, compared to 9 time slots for Q-CSMA and 27 time slots for the LLQ.

We also test our algorithms under medium and heavy traffic conditions, which poses increasing challenges to the learning method. The average packet delays across different traffic scenarios are depicted in Figure 4. Under medium traffic conditions, the MARL framework's shared and separate policies result in average delays of 1.59 and 1.41 time slots, respectively. The latter represent a 34.4% reduction compared to Q-CSMA (2.15) and a 59.0% reduction over LLQ (3.44). In the case of a heavy traffic condition experimented here, which is relatively close to the boundary of the capacity region, the LLQ benchmark algorithm experiences significant challenges, resulting in an average packet delay of up to 9.41 time slots. The Q-CSMA scheduler is still stable and results in average packet delay of 3.38 time slots. Both our MARL method achieves average delay that less than 2 time slots, a substantial 44% reduction compared to Q-CSMA.

In terms of the CDF, average delay, and maximum packet delay, the MARL methods demonstrate substantial improvements over the benchmarks. A close examination of the agents' policies reveals several key differences. Q-CSMA transmits more conservatively as it is designed to only schedule links in an independent set, thereby avoiding conflicts. In contrast, MARL adopts a more aggressive scheduling strategy. Since conflicts in our conflict graph are directional, MARL can schedule conflicting transmissions, ensuring that at least one link successfully transmits. In addition, compared with both benchmarks, our MARL method operates within a much richer action space.

We further validate the proposed MARL method on larger conflict graphs abstracted from Fig. 1a and Fig. 1b. In these scenarios, both separate and shared policies remain stable and continue to outperform the other two benchmark algorithms, as illustrated in Fig. 5 and Fig. 6. The results demonstrates that our method scales well with the network size.

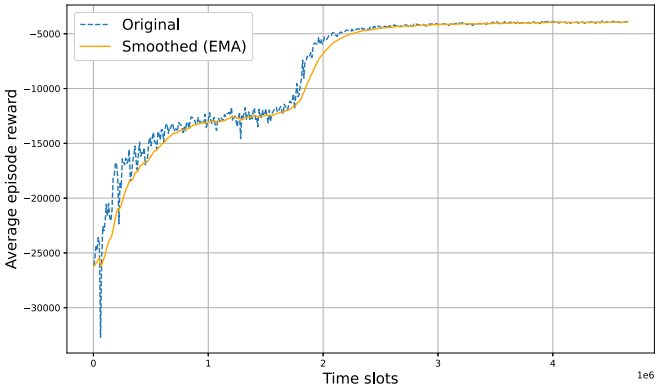


Fig. 7: Rewards of training episodes with shared policies.

D. Policy Convergence

With the 57-link 19-agent conflict graph abstracted from Fig. 1b, we test the learned policies once every five training episodes and plot the rewards. The average reward from the MARL method using a shared policy is depicted as the blue dashed line in Fig. 7, with clarity enhanced by an exponential moving average (EMA) curve in orange. Initially, the average

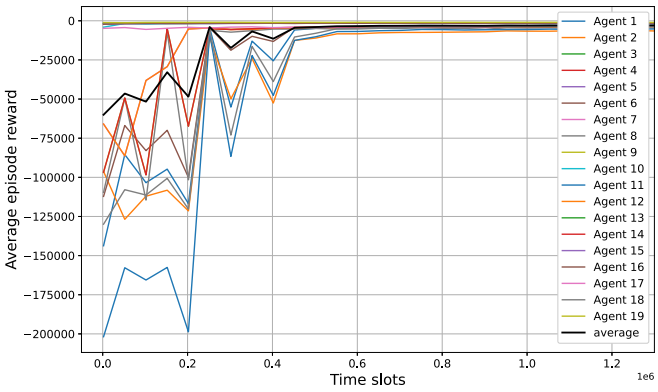


Fig. 8: Rewards of training episodes with separate policies.

Traffic load for testing:	Light	Medium	Heavy
if trained under light traffic:	good	mixed	unstable
if trained under medium traffic:	good	good	unstable
if trained under heavy traffic:	good	good	good

TABLE I: Training and testing mismatch.

reward improved rapidly, indicating quick learning by the agents. After approximately 1,000,000 time slots, the rewards plateaued, suggesting convergence to a sub-optimal solution. Despite the occasional fluctuations in rewards, which are characteristic of exploration in MARL methods, the reward generally improved and eventually stabilized after about two million time slots. This indicates that each agent has successfully acquired an effective and stable policy, resulting in a stable and good cumulative reward.

Using separate policies for the same graph, rewards for the 19 agents' rewards are plotted in Fig. 8. In the initial training stage of training, agents with fewer devices to manage, such as agent 3 (which serves only one device), and those with minimal conflict with neighboring agents, such as agents 5 and 7, quickly achieved good rewards. In contrast, agents like number 11, handling more devices, and agent 1, dealing with significant neighbor interference, encountered early challenges. Despite the fluctuations, all agents' reward trends upward in general, and they converge to efficient policies somewhat more rapidly than the shared policy approach, achieving convergence within approximately 600,000 time slots. These policies not only benefit the individual agents but also contribute to a stable and favorable cumulative reward for the entire network.

E. Model Mismatch

We also investigate the robustness of the MARL method when trained and tested under mismatched traffic conditions. We characterize the performance as "unstable" if the queue lengths are observed to generally increase persistently. Conversely, the performance is deemed "good" if it demonstrates satisfactory QoS performance when compared to the benchmark. The term "mixed" is used to describe a mixture of both "good" and "unstable" performance amongst the agents.

Table I demonstrates that policies trained under heavier traffic loads exhibit better performance when handling lighter traffic loads. For instance, policies trained in heavy traffic loads demonstrate satisfactory behavior in both light and medium traffic environments. Conversely, policies trained in light traffic loads show inadequate performance under medium and heavy traffic conditions.

V. Conclusion

We have introduced a novel multi-agent deep reinforcement learning framework for addressing the distributed scheduling problem in partially observable conflict graphs. The proposed method adopts a "modified" centralized training and distributed execution paradigm. The simulation results demonstrate the efficacy, scalability, and robustness of the trained policies. The proposed framework for conflict graphs can be potentially extended to signal-to-interference-based models as well as a broader set of resource allocation problems.

Acknowledgement

The authors thank Dr. Zhaoran Wang for suggesting the effective recurrent neural network structures and thank Drs. Shuping Yeh, Hosein Nikopour and Mark Eisen for useful comments.

References

[1] A. A. Al-Habob, O. A. Dobre, A. G. Armada, and S. Muhaidat, "Task scheduling for mobile edge computing using genetic algorithm and conflict graphs," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8805–8819, 2020.

[2] Z. Zhao, G. Verma, C. Rao, A. Swami, and S. Segarra, "Distributed scheduling using graph neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2021, pp. 4720–4724.

[3] R. Srikant and L. Ying, *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[4] R. E. Tarjan and A. E. Trojanowski, "Finding a maximum independent set," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 537–546, 1977.

[5] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 825–836, 2011.

[6] K. Shen and W. Yu, "Fractional programming for communication systems—part i: Power control and beamforming," *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2616–2630, 2018.

[7] K. Yang, D. Li, C. Shen, J. Yang, S.-p. Yeh, and J. Sydir, "Multi-agent reinforcement learning for wireless user scheduling: Performance, scalablility, and generalization," in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, 2022, pp. 1169–1174.

[8] O. Orhan, V. N. Swamy, M. Rahman, H. Nikopour, and S. Talwar, "Graph neural networks to enable scalable MAC for massive mimo wireless infrastructure," in *Proceedings of International Conference on Artificial Intelligence in Information and Communication*, 2023, pp. 489–494.

[9] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[10] F. H. Panahi, F. H. Panahi, and T. Ohtsuki, "Intelligent cellular offloading with VLC-enabled unmanned aerial vehicles," *IEEE Internet of Things Journal*, pp. 17 718–17 733, 2023.

[11] M. K. Tefera, S. Zhang, and Z. Jin, "Deep reinforcement learning-assisted optimization for resource allocation in downlink OFDMA cooperative systems," *Entropy*, vol. 25(3), no. 413, 2023.

[12] Y. Zhang and D. Guo, "Distributed MARL for scheduling in conflict graphs," in *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2023, pp. 1–8.

[13] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5872–5881.

[14] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[15] G. Qu, A. Wierman, and N. Li, "Scalable reinforcement learning of localized policies for multi-agent networked systems," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control, PMLR*, vol. 120, 2020, pp. 256–266.

[16] H. Wei, X. Liu, W. Wang, and L. Ying, "Sample efficient reinforcement learning in mixed systems through augmented samples and its applications to queueing networks," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 2033–2055.

[17] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[18] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.