Multi-Agent Reinforcement Learning for Multi-Cell Spectrum and Power Allocation

Yiming Zhang, *Student Member, IEEE*, and Dongning Guo, *Fellow, IEEE*

***Abstract*—Efficient and scalable radio resource allocation is essential for the success of wireless cellular networks. This paper presents a fully scalable multi-agent reinforcement learning (MARL) framework, where each agent manages spectrum, power allocation, and scheduling within a cell, using only locally available information. The objective is to minimize packet delays under stochastic traffic arrivals, applicable to both conflict graph models and cellular network configurations. This is formulated as a distributed learning problem and implemented using a multi-agent proximal policy optimization (MAPPO) algorithm. This traffic-driven MARL approach enables fully decentralized training and execution, ensuring scalability to arbitrarily large networks. Extensive simulations demonstrate that the proposed methods achieve quality of service (QoS) performance comparable to centralized algorithms that require global information, while the trained policies show robust scalability across diverse network sizes and traffic conditions.**

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

I. Introduction

The increasing density of devices and access points (APs) in cellular networks, driven by growing consumer demands, has heightened the significance of coordinated resource allocation between cells. In this context, the success of wireless cellular networks hinges on the ability of every AP in every cell to allocate resources efficiently, where decisions may involve which mobile device to serve in the downlink, at what time, using which subbands, and at what power levels. The goal of this work is to develop scalable, traffic-driven and *fully distributed* methods that achieve comparable quality of service (QoS) as those of well-known *centralized* methods, including the weighted minimum mean-squared error (WMMSE) [1] and fractional programming (FP) [2]. In particular, the proposed traffic-driven approach prioritizes packet latency as the primary QoS metric, with system dynamics explicitly incorporating queue length variations. By *fully distributed*, we refer to a system in which each AP executes an algorithm that requires input from other APs in at most a small neighborhood, so that a typical cell has fixed computational complexity even if the number of cells keeps increasing as the network expands.

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

This work was presented in part at the 60th Annual Allerton Conference on Communication, Control, and Computing and the 58th Asilomar Conference on Signals, Systems, and Computers in 2024.

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.

To better understand the challenges in resource allocation, we first abstract the wireless communication network as a conflict graph, which effectively represents interference and constraints between links. In a conflict graph, centralized methods like the max-weight algorithm achieve the optimal throughput [3], but require identifying all maximum independent sets within the graph, which is an NP-complete problem [4]. While greedy maximal scheduling (GMS) provides a simpler alternative, it remains centralized and thus impractical for large networks. Low-complexity heuristic methods, such as longest-queue-first (LQF) often achieve only a portion of the capacity region. As a more practical solution, the queue-length-based carrier-sense multiple access (Q-CSMA) [5] was proposed, offering improved performance over LQF while utilizing only local information.

Building on insights from conflict graphs, we extend to cellular networks with analog channel states, where resource allocation includes not only scheduling but also spectrum and power allocation. Existing approaches, however, have notable limitations. Centralized methods like WMMSE and FP require complete channel state information (CSI) across the entire network, with computational complexities that scale rapidly as network size increases. Heuristic methods, such as random/full-power allocation, require minimal information, but typically achieve only a fraction of the capacity region. ITLinQ [6], a low-complexity scheduling method, sequentially schedules transmissions in subsets of links with “sufficiently” low interference levels, which still requires global CSI and coordination. Distributed optimization approaches such as [7], [8] attempt to reduce the need for extensive information exchange, but they often underperform relative to centralized methods.

Is it possible to achieve QoS performance comparable to centralized approaches while only utilizing locally available information for decision-making? Machine learning has recently emerged as a powerful tool for wireless resource allocation problems, offering potential solutions to this challenge. Supervised learning approaches, as demonstrated in [9], have trained deep neural networks using WMMSE-generated datasets to approximate its policy. In [10], graph neural networks were adopted to leverage topological information for user scheduling in conflict graphs. Reinforcement learning (RL) offers advantages in avoiding high-dimensional, non-convex optimization, providing a model-free approach, and aligning well with sequential decision-making. Pioneering work applying deep RL to power control [11] achieved sum-rate performance closely matching that of FP and WMMSE algorithms. Further advancements have expanded RL applications to joint sub-band selection and transmit power control using deep Q-networks [12] and actor-critic networks [13]. Multi-agent RL (MARL) introduces multiple agents that interact and learn simultaneously to achieve desirable rewards, with applications in power allocation [14]–[16], scheduling [17] and beamforming [18].

The preceding learning methods fail to allow for a truly distributed deployment with limited observations at each access point. Supervised learning method [9] learns from WMMSE, which requires global CSI. The graph neural network approach [10] requires knowledge of the global topology. The methods in [11]–[13] require extensive CSI exchange between links. In [12], the use of Cartesian product action spaces also faces convergence issues as subbands increase. In [15] and [18], the authors make assumptions about independent transition functions and the reward is shared by all agents. The centralized training and distributed execution (CTDE) framework, common in these RL-based works, limits their scalability. While [19] incorporates federated learning with MARL to enable distributed training, it still requires centralized parameter reporting and achieves lower performance compared to centralized methods.

To develop a fully decentralized method, we frame resource allocation as a problem of decentralized partially observable Markov decision process with individual rewards (Dec-POMDP-IR). This framework accurately models the system dynamics in both conflict graphs and cellular networks. While a comprehensive theoretical study is beyond the scope of this work, we carefully refine the CTDE framework, adopting the multi-agent proximal policy optimization (MAPPO) algorithm with recurrent neural networks. In particular, our decentralized training and execution framework utilizes only locally available information during both training and execution phases, ensuring scalability. Extensive simulation results demonstrate the effectiveness and robustness of our proposed solutions across various network configurations.

Another key distinction of our work from previous studies [1], [2], [6]–[12], [15], [16], [18]–[20] lies in the choice of QoS metric. While previous work often focuses on maximizing sum-rate or throughput, we prioritize minimizing average packet delay for two reasons: First, traffic conditions are frequently lighter than the capacity allows, making delay a more relevant indicator of user experience. Second, maximizing sum-rate at the expense of certain links can lead to excessive delays for those links. With this focus on delay, we develop a traffic-driven MARL method, where states, rewards, and transitions are carefully designed around queue information. Our objective is to learn flexible, adaptable policies that map dynamic traffic and CSI to a wide array of actions. Unlike approaches that converge to static solutions [9]–[12], [15], [16], [18]–[20], our approach generalizes across traffic conditions, enabling the neural network to handle variations in traffic loads and channel conditions without retraining.

This paper makes several key contributions:

- We formulate traffic-driven resource allocation as a distributed learning problem within the Dec-POMDP-IR framework, incorporating partial observation, individual rewards, and local information sharing. We apply this formulation to conflict graphs and cellular networks, detailing the design of state spaces, action spaces, and reward functions.
- We adapt the conventional CTDE framework to a fully decentralized solution, ensuring that each agent's training cost and neural network size remain constant, regardless of the system's scale. This implementation leverages recurrent neural networks and MAPPO, with a detailed flowchart of the process and information exchange.
- We validate our solution's performance, scalability, and robustness through extensive simulations across diverse network configurations and traffic conditions. A full set of computer code is provided for reproducibility [21].

The paper is organized as follows: Sec. II formulates the learning problem and describes the conflict graph and cellular network models. Sec. III proposes two MARL-based solutions. Sec. IV discusses the simulation setup and numerical results. Sec. V provides concluding remarks.

II. MARL Framework and System Model

A. MARL framework

An *agent* processes information from the environment and makes decisions to obtain rewards. Consider multiple agents with indices from the set $\mathcal{K} = \{1, \ldots, K\}$ interacting with the environment over time slots $t = 1, \ldots, \mathcal{T}$, where $\mathcal{T}$ is the episode length. In time slot t:

- The environment is described by a global state $\mathbf{s}^{(t)}$,
- Agent k takes an action $a_k^{(t)}$ based on its *belief* of the environment, collectively forming a joint action $\mathbf{a}^{(t)} = \{a_1^{(t)}, \ldots, a_K^{(t)}\}$. The belief is derived on agent k's local observation $O_k^{(t)}$ of the environment, some additional observations shared by its neighbors, and the history of these observations, denoted as $\tau_k^{(t)}$.
- We assume a Markov model, where the probability of the next global state $\mathbf{s}^{(t+1)}$ is solely determined by the current state $\mathbf{s}^{(t)}$ and the joint action $\mathbf{a}^{(t)}$, which is denoted as
$$p\big(\mathbf{s}^{(t+1)} | \mathbf{s}^{(t)}, \mathbf{a}^{(t)}\big). \tag{1}$$
- At the end of each time slot, agent k receives an individual reward $R_k^{(t)}$.

We model the multi-agent problem as Dec-POMDP [22] with individual reward, which we refer to as Dec-POMDP-IR. Let $\mathcal{S}$ denote the state space in this framework, where the state at time t $\mathbf{s}^{(t)} \in \mathcal{S}$. Agent k's action space is denoted as $\mathcal{A}_k$, where $a_k^{(t)} \in \mathcal{A}_k$ represents the action taken by agent k at time t. The joint action space is defined as $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_K$. The transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ specifies the transition probability defined in (1). We also define the observation space for agent k as Ω_k and observation function as $\mathcal{O}$, which maps the state to the local observation $O_k \in \Omega_k$ for $k \in \mathcal{K}$. The reward function is defined as $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^K$, indicating that each agent receives an individual reward instead of a shared common reward in each transition. With discount factor γ balancing immediate and future rewards, our Dec-POMDP-IR can be represented by the tuple $\langle \mathcal{K}, \mathcal{S}, \{\mathcal{A}_i\}_{i \in \mathcal{K}}, \mathcal{P}, \mathcal{R}, \{\Omega_i\}_{i \in \mathcal{K}}, \mathcal{O}, \gamma \rangle$.

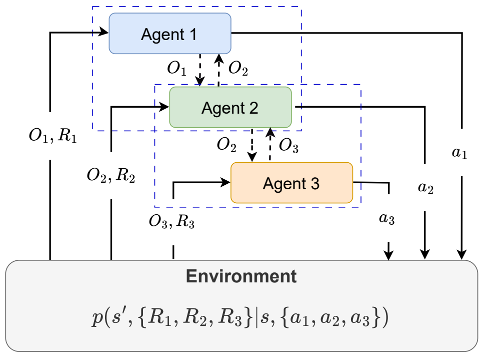


Fig. 1: Examples of Dec-POMDP-IR model with three agents.

Fig. 1 illustrates a Dec-POMDP-IR with three agents: Agents 1 and 2 form one neighborhood, and Agents 2 and 3 form another. Agents receive local observations from the environment and communicate with their neighboring agent(s). Subsequently, agents make decisions based on the entire history of their observations and receive rewards. The global state evolves based on joint actions and exogenous randomness.

The *policy* of agent k is denoted as π_k, which represents a conditional probability distribution of actions based on the agent k's belief. Agent k samples its action a_k from this distribution. The learning goal for agent k is to find a good *policy* π_k to maximize its own cumulative discounted reward:

$$\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_k^{(t)}\left(\mathbf{s}^{(t)}, \mathbf{a}^{(t)}, \mathbf{s}^{(t+1)}\right)\right]. \tag{2}$$

where the expectation $\mathbb{E}_{\pi}$ assumes that the initial state is sampled from the initial state distribution, each agent follows its policy π_k to select an action in each slot.

Our MARL problem formulation differs from other MARL problem formulations in resource allocation [12], [15], [18]. First, we do not explicitly define a local state space for each agent as even the network size and composition are unknown to individual agents. Second, we do not assume *transition independence* across agents, as agents' actions (allocation decisions) significantly impact other agents' observations and the evolution of the global state. Therefore, the Markovian property in (1) holds for the global state and joint action but not for individual agents. Third, we do not assume a common cooperative reward, as our goal is to develop a fully decentralized framework.

In Secs. II-B and II-C, we present two network models, for which we provide learning-based solutions in subsequent sections. The first model is simpler and the second model builds on the first one to describe a multi-cell wireless network with multiple frequency subbands.

B. Conflict Graph

1) System Model: The challenge of scheduling in conflict graphs involves allocating resources (e.g., radio spectrum subbands, computing units) to conflicting tasks or events. Consider a directed conflict graph denoted as $\mathcal{G} = (\mathcal{I}, \mathcal{E})$, where each vertex in $\mathcal{I}$ represents a device, and an edge $(i, j) \in \mathcal{E}$ with $i, j \in \mathcal{I}$ and $i \neq j$ indicates device i would cause conflict to device j if they are scheduled simultaneously.

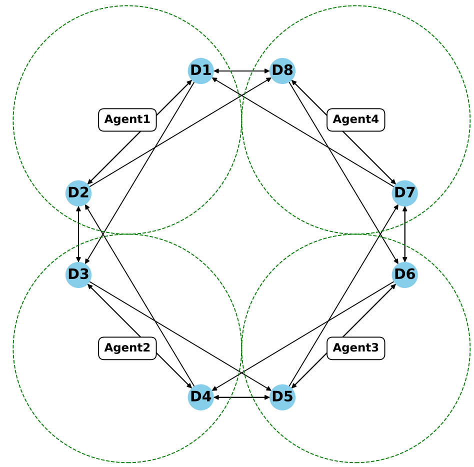


Fig. 2: A conflict graph of 4 agents in a symmetric deployment.

Fig. 2 depicts a conflict graph where each agent serves two devices. Each device operates a first-in-first-out (FIFO) queue for assigned tasks. Time is slotted, and each device receives a random number of new tasks at the beginning of each time slot. For simplicity and without loss of generality, all tasks require identical resources to proceed and agents have unit capacity, meaning one task can be successfully processed during one time slot if the device is scheduled. Upon successful processing, the task departs from the queue.

The directional edges in Fig. 2 indicate conflicts between devices. For example, if device 1 is scheduled, it would potentially cause conflict with device 2, 3 and 8. We adopt the standard collision model, where a task is successfully processed if and only if no other conflicting devices are scheduled in the same time slot. If a conflict occurs, the task processing fails, and the task remains in the queue.

2) Problem Formulation: We now formulate the scheduling problem in conflict graph as a Dec-POMDP-IR model. In a K-agent N-device conflict graph, where $\mathcal{K} = \{1, \dots, K\}$ and $\mathcal{N} = \{1, \dots, N\}$ denote the set of agent indices and device indices, respectively. We define $b_n \in \mathcal{K}$ as the serving agent of device n. Consequently, $\mathcal{N}_k = \{n \in \mathcal{N} \mid b_n = k\}$ denotes all the devices served by agent k. Fig. 2 depicts an example with $b_1 = 1$ and $\mathcal{N}_1 = \{1, 2\}$. At each time slot t, agent k makes a scheduling decision $a_k^{(t)}$, which is selected from: $\{0, 1, , \dots, |\mathcal{N}_k|\}$. A decision of 0 indicates that no device is scheduled by agent k during time slot t.

To ensure that our design is fully distributed, selected devices are represented by their local indices under each agent's control. For $k \in \{1, \dots, K\}$ and $\iota \in \{1, \dots, |\mathcal{N}_k|\}$, let $f(\iota, k)$ represent the global index of the ι-th (local) device served by agent k. For example, in Fig. 2, $f(1, 3) = 5$ and $f(2, 3) = 6$. The mapping f is a bijection.

Let $\mu_n^{(t)} \in \{0, 1\}$ denote the scheduling decision of device n in time slot t, i.e., $\mu_n^{(t)} = 1$ if agent b_n selects this device

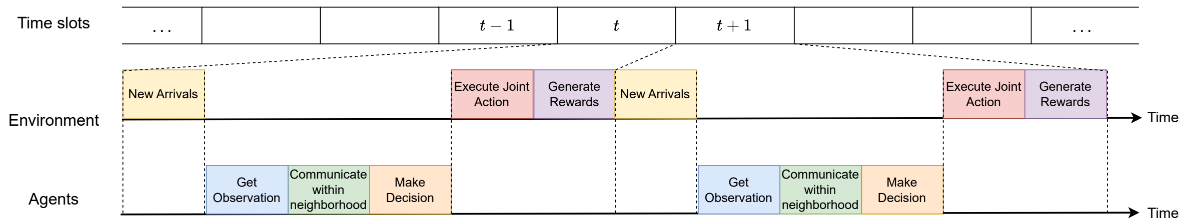


Fig. 3: Illustration of the timing of interactions between agents and environments.

(through its local index) in its action $a^{(t)}_{b_n}$. If a device is scheduled free from conflict with other devices, then one task is finished. The success or failure of device n in slot t is represented by

$$m^{(t)}_n = \begin{cases} 1, & \text{if } \mu^{(t)}_n = 1, \mu^{(t)}_i \neq 1 \text{ for all } (i,n) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Let $Y^{(t)}_n$ denote the number of newly arrived tasks to device n at the beginning of time slot t. The queue length of device n at the end of slot t can then be expressed as:

$$q^{(t)}_n = \max\left(0,\, q^{(t-1)}_n + Y^{(t)}_n - m^{(t)}_n\right). \quad (4)$$

We assume all queues start empty at $t = 0$. We also introduce the queue length after new arrivals as

$$\zeta^{(t)}_n = q^{(t-1)}_n + Y^{(t)}_n. \quad (5)$$

We can now express the global state $\mathbf{s}^{(t)} \in \mathcal{S}$ of the K-agent N-device conflict graph at each time slot t as:

$$\mathbf{s}^{(t)} = \left(\{q^{(t)}_n\}^N_{n=1}, \{\zeta^{(t)}_n\}^N_{n=1}\right). \quad (6)$$

With the scheduling decision of the agent k, $a^{(t)}_k$, defined in Sec. II-B2, we denote the joint action of all agents as $\mathbf{a}^{(t)} = \{a^{(t)}_1, \dots, a^{(t)}_K\}$. Based on the traffic dynamic described in (4), the transition from current global state $\mathbf{s}^{(t)}$ to next global state $\mathbf{s}^{(t+1)}$ is Markovian, as described by the state transition model (1).

Each agent can exchange information with agents within its neighborhood. Here we simply let agent k's neighborhood be defined to include all agents whose devices conflict with those served by agent k. For instance, in Fig. 2, 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 $\nu_{k,1}, \dots, \nu_{k,l_k}$ denote their indices. Let $\mathcal{C}(k) = \{k, \nu_{k,1}, \dots, \nu_{k,l_k}\}$ denote agent k's neighborhood, which always includes the agent itself. Agent k utilizes information from $\mathcal{C}(k)$ to make scheduling decisions for all devices it served.

The global indices of agent k's devices are $f(1, k)$, $\dots$, $f(|N_k|, k)$. The local observation of agent k in time slot t is defined to include local queue lengths as:

$$O^{(t)}_k = \left(\zeta^{(t)}_{f(1,k)}, \dots, \zeta^{(t)}_{f(|\mathcal{N}_k|,k)}\right). \quad (7)$$

The aggregate information of agent k in time slot t includes the local observations of agent k's neighboring agents and itself:

$$X^{(t)}_k = \left(O^{(t)}_k, O^{(t)}_{\nu_{k,1}}, \dots, O^{(t)}_{\nu_{k,l_k}}\right). \quad (8)$$

Each agent maintains a local observation history for a period of Υ time slots. The observation history in time slot t is defined as $\tau^{(t)}_k = \left(X^{(t-\Upsilon)}_k, \dots, X^{(t-1)}_k\right)$, which contains all information that is locally available to agent k at time t.

Since our goal is to minimize the delay, we define the learning objective using queue lengths as surrogates. Evidently, longer queue lengths lead to longer delays. Specifically, the direct contribution of agent k to the queue length objective can be expressed as:

$$u^{(t)}_k(\mathbf{s}^{(t)}) = -\sum_{i \in \mathcal{N}_k} q^{(t)}_i. \quad (9)$$

To promote collaborative behavior and encourage joint decisions that lead to mutually beneficial outcomes, we also incorporate the utilities of agent k's neighbors as indirect contributions. This approach discourages overly aggressive scheduling that might lead to frequent conflicts and performance degradation. Consequently, the individual reward function of agent k is defined as:

$$R^{(t)}_k\left(\mathbf{s}^{(t)}\right) = \sum_{i \in \mathcal{C}(k)} u^{(t)}_i. \quad (10)$$

Here the reward is simply a function of the current state, which is a special case of the more general form in (2). An crucial feature of our design is that the reward can be computed using only locally available queue length information. For example, the reward for agent 1 in Fig. 2 in slot t is equal to $-\big(q^{(t)}_1 + q^{(t)}_2 + q^{(t)}_3 + q^{(t)}_4 + q^{(t)}_7 + q^{(t)}_8\big)$, which agent 1 can compute using its own information and information from neighboring agents 2 and 4.

For clarity, we present the time flow of interactions between agents and environments in Fig. 3.

C. Cellular Network

1) System Model: We next define a wireless cellular network model that is more realistic than the conflict graph model. Indeed, the conflict graph (Fig. 2) is an abstraction of the cellular network deployment illustrated in Fig. 4. We restrict our attention to downlink transmissions in a cellular network comprising N (mobile) devices served by K access points (AP), one AP in each cell. All transmitters and receivers are equipped with a single antenna. As in Sec. II-B, $\mathcal{K} = \{1, \dots, K\}$ and $\mathcal{N} = \{1, \dots, N\}$ denote the set of cell indices and device indices, respectively. Each device $n \in \mathcal{N}$ is associated with its nearest AP, indexed as $b_n \in \mathcal{K}$. We refer to

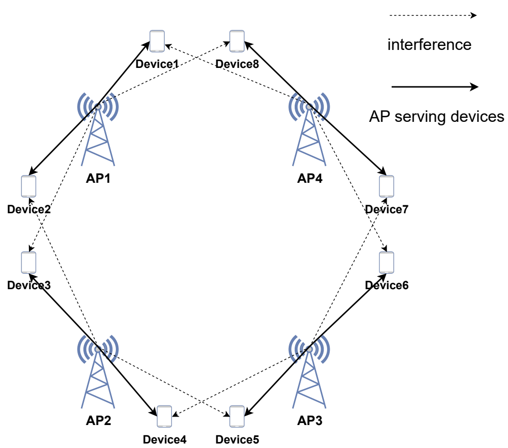


Fig. 4: A symmetric deployment with 4 APs and 8 devices.

the downlink from b_n to device n as link n. The set of devices served by AP k is denoted as $\mathcal{N}_k = \{n \in \mathcal{N} \mid b_n = k\}$.

Time is slotted with duration T, and the network utilizes H orthogonal subbands. The downlink channel gain from transmitter i to receiver j in time slot t on sub-band h is expressed as:

$$g_{i\to j,h}^{(t)} = \alpha_{i\to j} \left|\beta_{i\to j,h}^{(t)}\right|^2, \quad t = 1, 2, \dots \tag{11}$$

where $\alpha_{i\to j} \geq 0$ represents the large-scale path loss, which remains constant over many time slots. And $\beta_{i\to j,h}$ represents a small-scale Rayleigh fading component. In simulations, we use a first-order complex Gauss-Markov process to model small-scale fading: $\beta_{i\to j,h}^{(t)} = \rho\beta_{i\to j,h}^{(t-1)} + \sqrt{1-\rho^2}e_{i\to j,h}^{(t)}$ where $\left(\beta_{i\to j,h}^{(0)}, e_{i\to j,h}^{(1)}, e_{i\to j,h}^{(2)}, \dots\right)$ are independent and identically distributed circularly symmetric complex Gaussian random variables with unit variance.

The power allocated to transmitter n by its associated AP b_n in time slot t on sub-band h is denoted as $p_{n,h}^{(t)}$. Assuming additive white Gaussian noise with power σ^2 for all receivers across all subbands, the downlink spectral efficiency of link n in time slot t on sub-band h is:

$$C_{n,h}^{(t)} = \log\left(1 + \frac{g_{n\to n,h}^{(t)}\ p_{n,h}^{(t)}}{\sum_{j\in\mathcal{N},j\neq n} g_{j\to n,h}^{(t)}\ p_{j,h}^{(t)} + \sigma^2}\right). \tag{12}$$

Each AP acts as an agent, scheduling transmissions and allocating power for all devices within its cell. The neighborhood concept applies here as well, with agent k's neighborhood including all agents whose devices may cause sufficiently high interference to the devices in $\mathcal{N}_k$. Specifically, if the ratio of path loss gains $\alpha_{i\to i}/\alpha_{j\to i}$ is below a certain threshold, the device n is considered to be potentially highly interfered by devices in $\mathcal{N}_k$. Consequently, the neighborhood of agent k would include agent b_n.

For practical reasons, an AP cannot serve multiple links on the same sub-band simultaneously. In each time slot, the agent k needs to make scheduling decision and decide the transmission power $p_{k,h}^{(t)}$ for on each sub-band h:

$$z_{k,h}^{(t)} \in \{0, \dots, |\mathcal{N}_k|\}, \tag{13}$$

$$p_{k,h}^{(t)} \in \left\{P_{\min}, P_{\min}\left(\frac{P_{\max}}{P_{\min}}\right)^{\frac{1}{|\mathcal{P}|-1}}, \dots, P_{\max}\right\}. \tag{14}$$

A decision of $z_{k,h}^{(t)} = 0$ indicates that no links in cell k are activated during time slot t on sub-band h. Alternatively, an agent may select one of the links within its cell for transmission using transmission power from a quantized log-step power options ranging from $P_{\min}$ to power constraint $P_{\max}$. Links that are not selected remain silent in time slot t on sub-band h (i.e., power set to 0).

Without loss of generality, we assume identical packet size. Let L denote the packet size in bits, and W_h denotes the bandwidth of sub-band h. The queueing dynamics for each link with the queue length (in bits) of link n at the end of slot t expressed as:

$$q_n^{(t)} = \max\left(0,\ q_n^{(t-1)} + Y_n^{(t)}L - T\sum_{h=1}^{H} C_{n,h}^{(t)} W_h\right) \tag{15}$$

where $Y_n^{(t)}$ denotes the number of newly arrived packets to device n at the beginning of time slot t. The spectral efficiency is a function of decision variable $\left(z_{b_n,h}^{(t)}, p_{b_n,h}^{(t)}\right)$.

2) Problem Formulation: The cellular network problem can be formulated as Dec-POMDP-IR as well. Consider the global CSI that consists of all channel gains between all transmitters and receivers across all subbands:

$$\mathbf{G}^{(t)} = \left\{g_{i\to j,h}^{(t)} \mid i,j \in \{1,\dots,N\}, h \in \{1,\dots,H\}\right\}. \tag{16}$$

At each time slot t, the global state $\mathbf{s}^{(t)} \in \mathcal{S}$ of the K-agent N-device cellular network is given by:

$$\mathbf{s}^{(t)} = \left(\mathbf{G}^{(t)}, \{q_n^{(t)}\}_{n=1}^N, \{\zeta_n^{(t)}\}_{n=1}^N\right) \tag{17}$$

where $\zeta_n^{(t)} = q_n^{(t-1)} + Y_n^{(t)}L$ represents the queue length of device n after receiving new packet arrivals.

The action of agent k in time slot t is defined as $a_k^{(t)} = \left(z_{k,h}^{(t)}, p_{k,h}^{(t)}\right)_{h=1}^H \in \mathcal{A}_k$, indicating the device selection and corresponding power level across all subbands. The Markov state transition model for the cellular network follows that of the conflict graph, with the Markovian transition probability from the current global state $\mathbf{s}^{(t)}$ to the next global state $\mathbf{s}^{(t+1)}$ defined as $p\left(\mathbf{s}^{(t+1)}|\mathbf{s}^{(t)}, \mathbf{a}^{(t)}\right)$.

We assume that for each device n, the transmitter learns the direct channel gain $g_{n,h}$ on sub-band h via receiver feedback, while the receiver measures the total interference-plus-noise power and its spectral efficiency. Both transmitters and receivers report their own CSI to the corresponding agent (cell) b_n, delayed by one time slot. The transmitter also records the transmission power from previous time slot. Additionally, we assume that each agent has timely queue length information for all links within its cell. Therefore, the locally available information $o_n^{(t)}$ for device n in time slot t includes:

- $\zeta_n^{(t)}$: the queue length of device n;

- $g_{n\to n,h}^{(t-1)}$, $h=1,\ldots,H$: the direct gains on all subbands;
- $p_{n,h}^{(t-1)}$, $h=1,\ldots,H$: device n's actions;
- $\sum_{j\in\mathcal{N},j\neq n} g_{j\to n,h}^{(t-1)}\ p_{j,h}^{(t-1)}+\sigma^2$, $h=1,\ldots,H$: the interference-plus-noise power levels at device n's receiver on all subbands;
- $C_{n,h}^{(t-1)}$, $h=1,\ldots,H$: spectral efficiencies of link n computed from (12) on all subbands.

As in the conflict graph, we employ a local-to-global index mapping strategy to ensure a fully distributed design. The local observation of agent k in time slot t, denoted by $O_k^{(t)}$, is defined as:

$$O_k^{(t)}=\left(o_{f(1,k)}^{(t)},\ldots,o_{f(|\mathcal{N}_k|,k)}^{(t)}\right), \tag{18}$$

which includes delayed CSI information, delayed action and timely queue length information of all links within its cell. Then the local aggregate information of agent k in time slot t is $X_k^{(t)}=\left(O_k^{(t)},O_{\nu_{k,1}}^{(t)},\ldots,O_{\nu_{k,l_k}}^{(t)}\right)$ and the observation history in time slot t is $\tau_k^{(t)}=\left(X_k^{(t-\Upsilon)},\ldots,X_k^{(t-1)}\right)$.

The reward function for the cellular network model is defined analogously to that of the conflict graph model, aiming to minimize packet delay using queue lengths as surrogates. The direct and indirect contributions to the queue length objective, as well as the individual reward function for each agent, are calculated using the same formulas (9) and (10) where the units of queue lengths are now bits.

Formulating the cellular network and conflict graph as a Dec-POMDP-IR captures the essence of decentralized decision-making under partial observability and constrained information sharing, with individual rewards for each agent. The primary objective for each agent k is to devise an optimal policy π_k that effectively maps the local aggregate information X_k in each time slot to a strategic sequence of actions a_k. This policy serves a dual purpose: 1) to maximize the agent's own reward function, as defined in (10), and 2) as a consequence, to minimize overall packet delay within the network.

III. MARL-based solution

This section presents our MARL-based solution to the Dec-POMDP-IR problem formulated earlier. We aim to find good control policies that yield desirable rewards within the constraints of decentralized decision-making and partial observability. The policy π_k of agent k is determined by a policy network parameterized by θ_k (denoted as π_{θ_k} in this section), which maps the local aggregate information $X_k^{(t)}$ and its history $\tau_k^{(t)}$ to a categorical distribution over discrete actions. The value network, parameterized by ϕ_k, estimates the expected return from a given state based on local aggregate information $X_k^{(t)}$ and its history $\tau_k^{(t)}$.

The policies are trained in parallel using trajectories of states, actions, and rewards. Our approach refines the popular on-policy training algorithm MAPPO, which has demonstrated success in various cooperative multi-agent tasks [23]. We adapt this algorithm to our specific setting and incorporate recurrent neural networks to process historical information effectively. Based on this framework, we propose two distinct training methods, each with its own strengths and trade-offs. First we describe the important recurrent neural network structures in the network.

A. Recurrent Neural Network

We incorporate recurrent neural network structures, specifically long short-term memory (LSTM) units, into both the policy and value networks for the following reasons: 1) While the Markovian property in (1) holds for the global state and joint action, from each agent's perspective, the local dynamics of $\left(X_k^{(t)},a_k^{(t)},r_k^{(t)},X_k^{(t+1)}\right)$ are non-Markovian. This is because the system dynamics include both the environment and other agents' continuously evolving policies, making the system non-stationary from a single agent's viewpoint. Therefore, making decisions based on information from a single time slot is insufficient. To capture this evolving nature of the environment and other agents' policies, we employ recurrent structures that process both current observations and historical information. 2) To make decisions based on historical information, directly inputting all historical data $\tau_k^{(t)}$ can be redundant and increase network size. LSTM layers can carry important information through a cell state and discard redundant information.

Both the policy network and value network contain two parts: an LSTM layer for history embedding and a multi-layer perceptron (MLP) for decision making/value estimation. We define the recurrent state of the policy network for agent k in time slot t to be $\hat{X}_k^{(t)}$, which serves as a compact representation of the history $\tau_k^{(t)}$. Similarly, we define the recurrent state of the value network for agent k in time slot t as $\tilde{X}_k^{(t)}$. By utilizing a recurrent architecture, agents can capture temporal dependencies and adapt to environment dynamics beyond a single observation, which allows agents to better infer neighboring agents' behaviors and impacts, making informed decisions based on augmented context. This approach also encourages consideration of both immediate and long-term effects in the decision-making process.

For the policy network of agent k in time slot t, the input includes local aggregate information $X_k^{(t)}$ and the previous time slot's recurrent state $\hat{X}_k^{(t-1)}$. The output includes the recurrent state of this time slot $\hat{X}_k^{(t)}$ (generated by the LSTM layer) and the action decision (generated by the MLP). The recurrent state is updated iteratively across time slots and carries important information as a historical embedding. The agent makes decisions based on the current time slot's local aggregate information and this embedding. Similarly, the value network takes $X_k^{(t)}$ and $\tilde{X}_k^{(t-1)}$ as input and outputs a value estimation and $\tilde{X}_k^{(t)}$.

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. Next we discuss specifics of two MARL-based solutions.

B. Individual Policies

Our first method implements a fully distributed approach for both training and execution. Inspired by the scalable framework in [24], we modify the typical CTDE process. Each agent k maintains its own policy and value networks and both input only local information $X_k^{(t)}$ and recurrent state. Specifically, the input to policy network is defined as $\hat{\mathcal{X}}_k^{(t)} = \left(X_k^{(t)}, \hat{X}_k^{(t-1)}\right)$, the input to value network is defined as $\tilde{\mathcal{X}}_k^{(t)} = \left(X_k^{(t)}, \tilde{X}_k^{(t-1)}\right)$. By limiting the neighborhood size, we ensure that network input dimensions remain constant regardless of the total number of agents in the system. This design enables truly decentralized operations, as each agent operates independently, managing its own trajectory, sampling from it, and training its networks to maximize its individual reward R_k. The resulting method is highly scalable and practical for large-scale networks.

For simplicity and formula reusability, we describe the decentralized training and execution process for agent k without carrying the sub-index k in the following formulas. Throughout this subsection, the reward R, policy network θ, value network ϕ, policy network input $\hat{\mathcal{X}}$, value network input $\tilde{\mathcal{X}}$, action a and sample batch B refer to the corresponding variables of agent k.

The training process is iterative, with both the policy and value networks being updated for a fixed number of slots after each episode. The networks from the previous training iteration are denoted as θ_{old} and ϕ_{old}. We first estimate the advantage function by the truncated version of generalized advantage estimation (GAE) in [25, Eq. 16] based on episode trajectory, for each time slot $t \in \{1, \dots, \mathcal{T}-1\}$:

$$A^{(t)} = \sum_{l=0}^{\mathcal{T}-t-1} (\gamma\lambda)^l \left(R^{(t+l)} + \gamma V_\phi(\tilde{\mathcal{X}}^{(t+l+1)}) - V_\phi(\tilde{\mathcal{X}}^{(t+l)})\right) \tag{19}$$

and $A^{(\mathcal{T})} = V_\phi(\tilde{\mathcal{X}}^{(\mathcal{T})})$, where $\lambda \in [0,1]$ is the GAE weighting parameter that controls the bias-variance trade-off along with discount factor γ.

After computing the advantage function for all time slots in the trajectory, we sample a batch of transitions from the trajectory with size $|B|$, where B stands for the sample of time indices. The sampled policy network input $\hat{\mathcal{X}}$, value network input $\tilde{\mathcal{X}}$, actions a and corresponding advantages A are used to update the networks. The value network parameters are updated to fit the estimated advantage values by minimizing the following loss function:

$$\begin{aligned}\mathcal{L}(\phi, \tilde{\mathcal{X}}, A) = \frac{1}{|B|}\sum_{t\in B} \max \Bigg[&\left(V_\phi\left(\tilde{\mathcal{X}}^{(t)}\right) - A^{(t)}\right)^2, \\ &\left(c_\epsilon\left(V_\phi\left(\tilde{\mathcal{X}}^{(t)}\right), V_{\phi_{\text{old}}}\left(\tilde{\mathcal{X}}^{(t)}\right)\right) - A^{(t)}\right)^2 \Bigg]\end{aligned} \tag{20}$$

where $c_\epsilon(x, y) = \min(\max(x, y-\epsilon), y+\epsilon)$ is a clipping function.

We define the probability ratio:

$$r_\theta(\hat{\mathcal{X}}, a) = \pi_\theta\left(a \mid \hat{\mathcal{X}}\right) \Big/ \pi_{\theta_{\text{old}}}\left(a \mid \hat{\mathcal{X}}\right). \tag{21}$$

Let $\mathcal{H}(\cdot)$ denote the Shannon entropy of a probability mass function and δ to be the entropy coefficient hyper-parameter. We update the policy network of agent k to maximize the objective function:

$$\begin{aligned}J(\theta, \hat{\mathcal{X}}, A) = &\frac{1}{|B|}\sum_{t\in B} \min\left(r_\theta^{(t)} A^{(t)}, c_\epsilon\ \left(r_\theta^{(t)}, 1\right) A^{(t)}\right) \\ &+ \delta \frac{1}{B}\sum_{t=1}^{B} \mathcal{H}\left(\pi_\theta\left(\,\cdot\,|\hat{\mathcal{X}}^{(t)}\right)\right).\end{aligned} \tag{22}$$

The combined policy and value networks constitute an actor-critic architecture, which generally enhances sample efficiency and accelerates convergence compared to actor-only (e.g., policy gradient) or critic-only (e.g., Q-learning) methods. In stochastic environments, trajectories can yield varying returns (defined as the discounted sum of future rewards from a given state), resulting in high variance when using returns directly as an objective function for policy network. While increasing batch size can mitigate this variance, it compromises sample efficiency. The value network, employing temporal difference learning for bootstrapping as evidenced in (19), provides more accurate return estimates. This approach reduces variance in the advantage function $A^{(t)}$, which is central to both the value network loss function in (20) and the policy network objective function in (22). Consequently, this formulation facilitates faster convergence and improved stability in the learning process.

C. Decentralized Training and Execution

The training process for agents with individual policies is summarized in Algorithm 1 and illustrated in Fig. 5. The process comprises two phases: data collection (indicated by solid lines in Fig. 5) and networks update (indicated by dash lines in Fig. 5). During the data collection phase, each agent k operates under its current policy for an episode. Throughout this episode, the agent k communicates local observation $O_k^{(t)}$ with neighboring agents and aggregates local information $X_k^{(t)}$. Provided with recurrent state from previous time slot, agent executes actions based on its policy $\pi_{\theta_k}\left(a_k^{(t)}|\hat{\mathcal{X}}_k^{(t)}\right)$. The agent records transitions, including local aggregate information $X_k^{(t)}$, recurrent state $\hat{X}_k^{(t)}$ (together forming input to policy network $\hat{\mathcal{X}}_k^{(t)}$), actions $a_k^{(t)}$, and rewards $R_k^{(t)}$. Subsequently, it computes advantage values $A_k^{(t)}$ retrospectively for each time slot using (19) and records value network recurrent state $\tilde{X}_k^{(t)}$ (which, with local aggregate information, forms input to value network $\tilde{\mathcal{X}}_k^{(t)}$). All this information is then stored in the agent's experience replay buffer.

The network update phase involves iterative refinement of the agents' policy and value networks. Each agent samples batch data $\left(X_k^{(t)}, \hat{\mathcal{X}}_k^{(t)}, \tilde{\mathcal{X}}_k^{(t)}, a_k^{(t)}, A_k^{(t)}\right)_{t\in B_k}$ from its replay buffer. The value network parameters ϕ_k are updated to minimize the critic loss function defined in (20), while the policy network parameters θ_k are adjusted to maximize the objective function given in (22). This process is iterated for a predetermined number of episodes and iterations, facilitating policy improvement based on accumulated experience. Once the training is finished, only the policy network is employed during execution (indicated by green lines in Fig. 5). This design ensures decentralized training and execution.

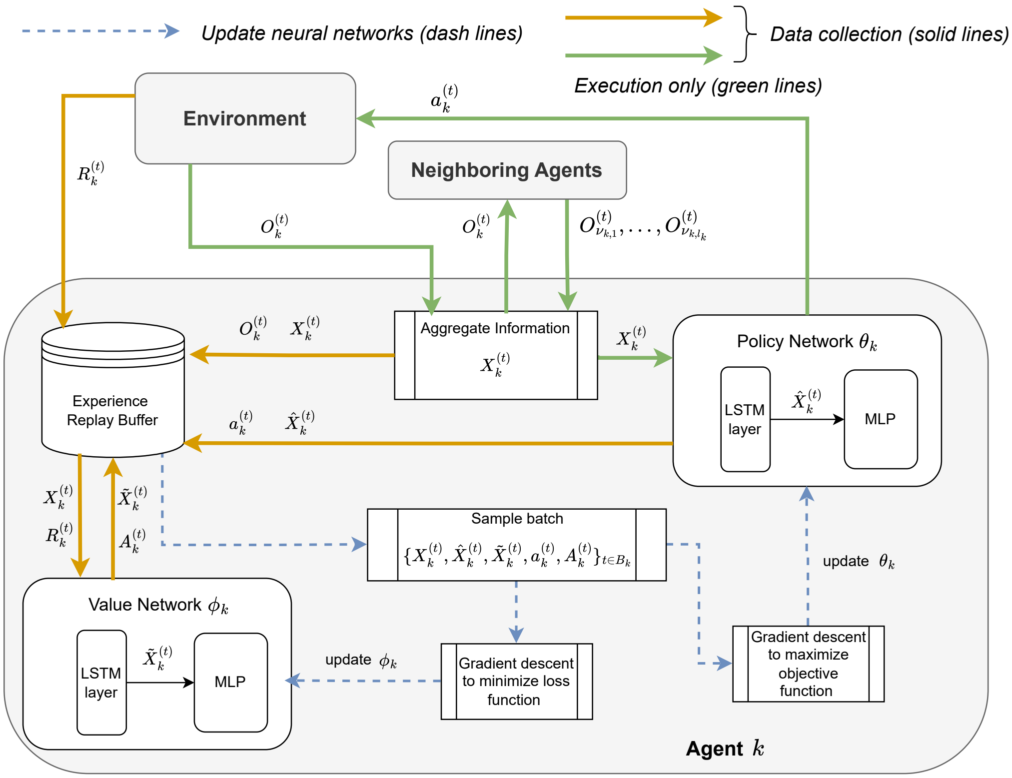


Fig. 5: Diagram of the training and execution workflow.

Algorithm 1 Decentralized training for agent k.

Initiate policy network θ_k and value network ϕ_k, initialize recurrent state $\hat{X}_k^{(0)}, \tilde{X}_k^{(0)}$
for each episode $e = 1, \ldots, E$ **do**
  /* Interact with environment and collect data */
  for time slot $t = 1, \ldots, \mathcal{T}$ **do**
    Communicate local information $O_k^{(t)}$ with neighboring agents.
    Take action based on $\pi_{\theta_k}\left(a_k^{(t)} | X_k^{(t)}, \hat{X}_k^{(t-1)}\right)$
    Record $\left(X_k^{(t)}, \hat{X}_k^{(t)}, a_k^{(t)}, R_k^{(t)}\right)$ to experience replay buffer.
  end for
  for time slot $t = 1, \ldots, \mathcal{T}$ **do**
    Compute advantages $A_k^{(t)}$ using (19).
    Record $\left(A_k^{(t)}, \tilde{X}_k^{(t)}\right)$ in experience replay buffer.
  end for
  /* Update policy and value networks */
  for iteration $n = 1, \ldots, N_{iteration}$ **do**
    $\phi_{k,old} \leftarrow \phi_k, \theta_{k,old} \leftarrow \theta_k$
    Take $\{X_k^{(i)}, \hat{X}_k^{(i)}, \tilde{X}_k^{(i)}, a_k^{(i)}, A_k^{(i)}\}_{i \in B_k}$ as a sample batch from the experience replay buffer.
    Update ϕ_k to minimize (20).
    Update θ_k to maximize (22).
  end for
end for

D. Shared Policy

The second method employs a partially decentralized framework. While both policy and value networks still use only local information $\hat{X}_k$, all agents share a common policy and value networks, and optimize a common collective reward using shared trajectories. Compared with first method, this approach allows the shared policy to benefit from the experiences of all agents during training, and it is more effective when computation resource is limited.

The training process is similar to that of the individual policies method, but with the loss function for shared critic function be

$$\mathcal{L}'(\phi, \hat{X}, A) = \frac{1}{K} \sum_{k=1}^{K} L(\phi, \hat{X}_k, A_k) \tag{23}$$

and the objective function for shared policy network be:

$$J'(\theta, \hat{X}, A) = \frac{1}{K} \sum_{k=1}^{K} J(\theta, \hat{X}_k, A_k). \tag{24}$$

IV. Simulation Results and Analysis

A. Simulation Setup

To evaluate the performance of proposed methods, we conducted simulations on both conflict graph and cellular network models under varying traffic intensities. In all scenarios, we let the number of packet arrivals to agent n in time slot t, denoted by $Y_n^{(t)}$, be an independent Poisson random variable with

This article has been accepted for publication in IEEE Transactions on Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2025.3534565

TABLE I: Cellular network parameters

Cell radius:	500m
Path loss (LTE standard):	$128.1 + 37.6 \log_{10}(\text{distance})$ (dB)
AWGN power:	$\sigma^2 = -114$ dBm
Max transmitter power:	$P_{max} = 23$ dBm
Discretized power levels:	$\|\mathcal{P}\| = 6$
Time slot duration:	$T = 20$ ms
Bandwidth for each sub-band:	$W_h = 20$ MHz
packet length	$L = 0.5$ Mbits

mean λ_n. Throughout this section, we assume the spectrum is divided into $H = 3$ subbands. Three distinct network configurations are considered:

1) The conflict graph depicted by Fig. 2, which is an abstraction of the downlink of the 8-device, 4-AP symmetrical deployment in Fig. 4.
2) A cellular network deployed as compact hexagons, consisting of 19 APs, with each AP serving 3 devices, as depicted in Fig. 6a.
3) A randomly deployed cellular network with 19 APs and 57 devices, as illustrated in Fig. 6b. Each device is associated to its nearest AP. An AP serves between 2 and 5 devices.

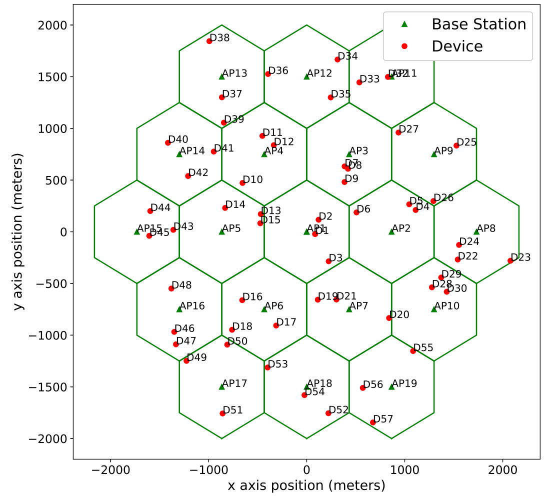
The cellular network simulations were conducted using the parameters listed in Table I.

To comprehensively evaluate our proposed MARL-based scheduler, we compared its QoS performance against several benchmark schemes across both the conflict graph and cellular network scenarios. For the conflict graph setting, we employed three benchmark schemes:

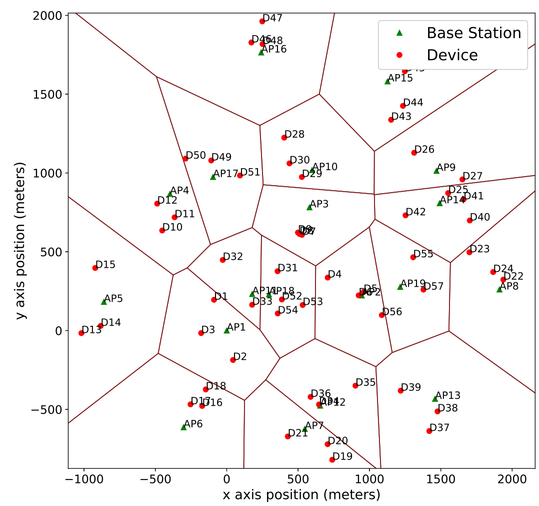
1) **GMS:** A centralized method that starts with an empty schedule, iteratively selects device with the longest queue in the network, adding it to the schedule and disabling those conflicting devices. This process repeats among the remaining devices until all devices are either scheduled or disabled.
2) **LLQ:** A distributed greedy method in which each AP schedules a device for transmission if it has a longer queue than all devices it has a conflict with; in case of a tie between j devices, each of those devices is scheduled independently with probability $1/j$.
3) **Q-CSMA [5]:** A method where devices perform carrier sensing prior to transmission, ensuring all scheduled devices form an independent set, and then each enabled device transmits with a certain probability based on its queue length.

For the cellular network scenarios, we utilized four benchmark schemes:

1) **LLQ:** A greedy method where APs use all subbands at full power to serve the device with the longest queue in their neighborhood. In case of a tie between multiple devices, one device is selected uniformly at random.
2) **ITLinQ [6]:** The APs use full power and all subbands to serve subsets of devices with "sufficiently" low interference between them based on the CSI. We actually simulate a slightly more complex version called Fair-ITLinQ [6], as the original ITLinQ exhibits poor performance in the 57-device scenario.



(a)



(b)

Fig. 6: Two networks with 19 cells serving 57 devices. (a) A regular deployment; (b) a random deployment.

3) **FP [2]:** A centralized iterative method based on minorization-maximization, assuming real-time global CSI is available. Device weights are proportional to queue lengths, and the subbands are allocated independently based on their respective CSI. To the best of our knowledge, the genie-aided FP method is essentially the best-performing resource allocation scheme, which performs similarly or outperform competitive techniques reported in [9], [11], [13], [15], [18], [20].
4) **WMMSE [1]:** A centralized iterative optimization algorithm, also assuming real-time global CSI availability. Device weights are proportional to queue lengths, and the subbands are allocated independently based on CSI.

This article has been accepted for publication in IEEE Transactions on Communications. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2025.3534565

TABLE II: MARL learning parameters for the policy and value networks.

Network optimizer	RMSprop for all neural networks
Learning rates	0.0001 for all neural networks
Number of recurrent layers	1 for all neural networks
Number of hidden layers	2 for all neural network
Neurons per hidden layer	64
Discount factor	0.995
Entropy coefficient	δ = 0.01
GAE parameter	λ = 0.95
Recurrent sequence length	64

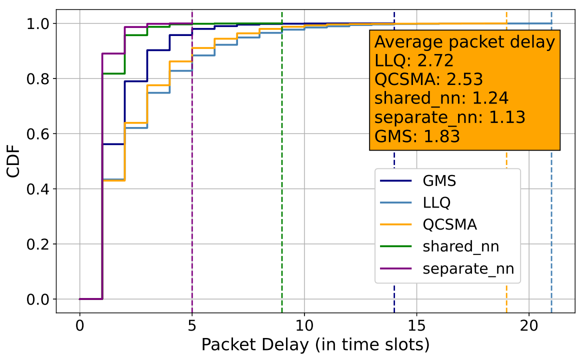


Fig. 7: CDFs of packet delays in the 8-device conflict graph.

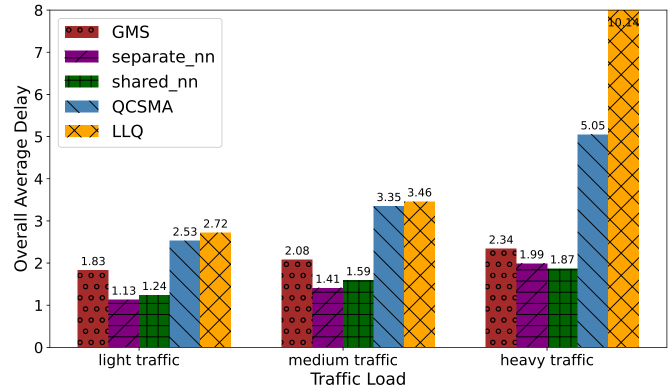


Fig. 8: The average packet delay of the 8-device conflict graph.

TABLE III: Information needed for different methods in a conflict graph.

Methods	**Queue length**	**Broadcast**
GMS	global	broadcast
LLQ	local	None
Q-CSMA	local	broadcast
MARL shared	local	None
MARL separate	local	None

Like FP, WMMSE is guaranteed to converge to a local optimum of the problem and offers comparable performance.

B. Training and Testing

We implemented both centrally trained shared policy and individually trained separate policies in Sec. III-B and III-D. Each training episode lasted 2,000 time slots. To prevent the scheduler from being trapped in adverse queueing conditions before adequate training, episodes were terminated and restarted if any device's queue length exceeded a predefined threshold. For testing or deployment, episodes spanned 5,000 time slots, with the average packet delay serving as the performance metric when the queue is considered stable. Other MARL learning parameters are summarized in Table II.

C. Performance Analysis

Our analysis encompasses both the conflict graph and the cellular network environments, providing insights into the effectiveness of our MARL-based approach across different network configurations.

1) QoS Performance in Conflict Graph: We first examine the conflict graph scenario, where we compare the average packet delays achieved by our MARL method against the GMS, LLQ and Q-CSMA benchmarks. In this context, packet delay is measured in time slots, assuming successful transmission of one packet per time slot using one sub-band when scheduled conflict-free.

Fig. 7 presents the cumulative distribution functions (CDFs) of packet delays under light traffic conditions in the conflict graph depicted by Fig. 2. Both MARL approaches—utilizing shared or separate policies—outperform the benchmarks, with their CDFs dominating those of the other methods. Notably, over 80% of packets are transmitted within a single time slot using either MARL method. The average packet delays of both MARL methods are lower than that of GMS and substantially lower than those of Q-CSMA and LLQ. Furthermore, MARL with separate policies achieves a significantly lower maximum packet delay (indicated by the dashed vertical line in Fig. 7) compared to GMS, Q-CSMA, and LLQ.

We further test our algorithms under medium and heavy traffic conditions, which pose increased challenges to the learning method. Fig. 8 illustrates the average packet delays across these scenarios. Under medium traffic conditions, the MARL-based solutions continue to outperform the benchmarks. In heavy traffic condition, which is relatively close to the boundary of the capacity region, the LLQ algorithm experiences high delays. The Q-CSMA scheduler's performance also degrades rapidly, while GMS remains stable and results in average packet delay of 2.34 time slots, whereas both MARL methods achieve average delays under 2 time slots.

The MARL methods demonstrate substantial improvements over benchmarks in terms of CDF, average delay, and maximum packet delay. A closer examination of the agents' policies reveals key differences: GMS and Q-CSMA transmit more conservatively, scheduling only devices in independent sets to avoid conflicts. In contrast, MARL methods operate in a richer action space, sometimes scheduling more aggressively than independent sets. Since conflicts occur between directional links, a subset of MARL-scheduled conflicting transmissions may still succeed.

2) Accessible Information and Time Complexity: Table III summarizes the required information for different methods in the conflict graph setting. GMS is centralized and requires

TABLE IV: Comparison of information exchange and execution time across resource management methods.

methods	queue lengths	CSI	execution time $N = 57, K = 19, H = 3$
FP	global	global	58.131 ms
WMMSE	global	global	926.882 ms
FITLinQ	local	global	5.223 ms
Greedy	local	None	0.166 ms
MARL shared	local	local	1.569 ms
MARL separate	local	local	1.303 ms

global queue length information. Q-CSMA and FITLinQ necessitate some centralized coordination as each link sequentially sends a broadcasting signal to decide whether to participate in transmission. Table IV outlines the required information and execution times of the various methods in cellular network scenarios. The execution times are measured as the average time per execution over a testing episode in the network depicted by Fig. 6a using a 4-core 2.8 GHz Core i7-1165G7 processor.

3) QoS Performance in Cellular Network: We now evaluate our algorithm in more complex cellular network scenarios. Delays are measured in milliseconds, and the number of bits delivered in each transmission is determined by the SINR, generally not an integer number of packets. The packet delay is calculated once all of its bits are received. To validate the scalability, we test our MARL methods on relatively large networks consisting of 19 APs and 57 devices, as shown in Figs. 6a and 6b. As traffic intensity increases, benchmarks using local information degrade quickly, while our separate and shared policies remain stable and continue to outperform them, as illustrated in Figs. 9 and 10. Compared to the genie-aided centralized methods, our methods offer similar performance in the network depicted by Fig. 6a and slightly better performance than FP and WMMSE in the network depicted by Fig. 6b. We plot the CDFs of packet delays for all successfully transmitted packets. As shown in Fig. 11, the CDFs of our two methods clearly dominates the CDFs of the other 4 benchmarks.

Our simulation results demonstrate that the proposed fully distributed MARL-based methods, using only local information, can achieve performance levels comparable to genie-aided centralized methods like FP and WMMSE. While centralized methods can achieve better results in some challenging scenarios by leveraging global information, our distributed approach trades off some performance for practical implementation with only local information. Notably, as traffic-driven approaches, our MARL-based solutions offer significant advantages in terms of real-time implementation. Table IV shows that the execution time for our MARL methods is approximately 1-2 milliseconds, which is substantially smaller than the observed packet delays. This rapid execution enables real-time decision-making in dynamic network environments. In contrast, FP and WMMSE, being iterative optimization-based methods, require more and often unpredictable computational resources. Their execution times are one to two orders of magnitude larger than our MARL-based methods, making them challenging to deploy in real-time systems where rapid adaptation to changing network conditions is crucial.

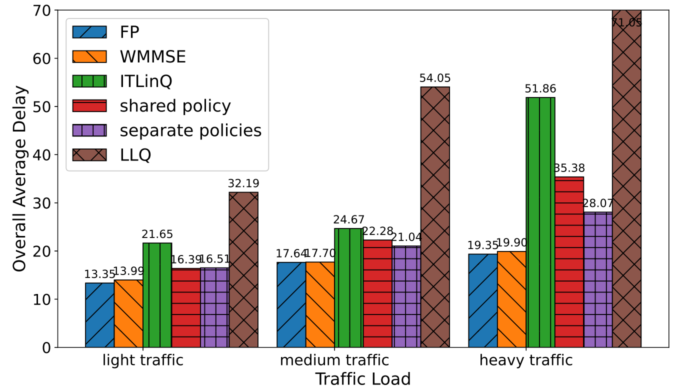


Fig. 9: QoS of the regular network depicted by Fig. 6a.

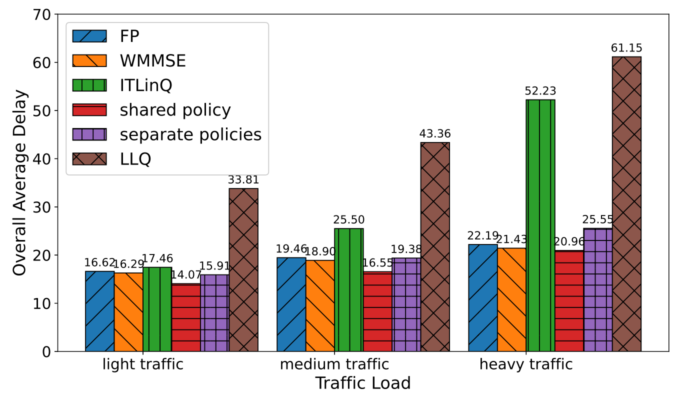


Fig. 10: QoS of the random network depicted by Fig. 6b.

Analysis of the agents' policies reveals that during training, they aim to balance utilizing as many subbands as possible for devices with long queues while avoiding excessive interference with neighbors based on local information.

D. Policy Convergence

To evaluate the training performance and algorithm convergence, we tested the learned policies every five training episodes and plotted the rewards. Fig. 12 illustrates the average

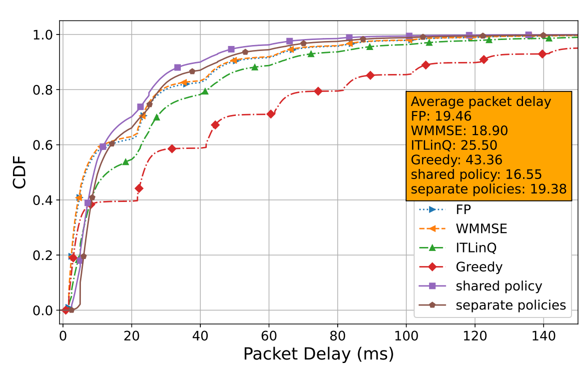


Fig. 11: CDFs of delays in the network depicted by Fig. 6b.

This article has been accepted for publication in IEEE Transactions on Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2025.3534565

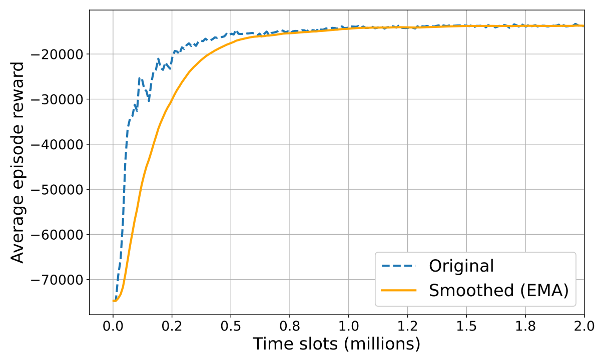


Fig. 12: Rewards of training episodes with shared policies.

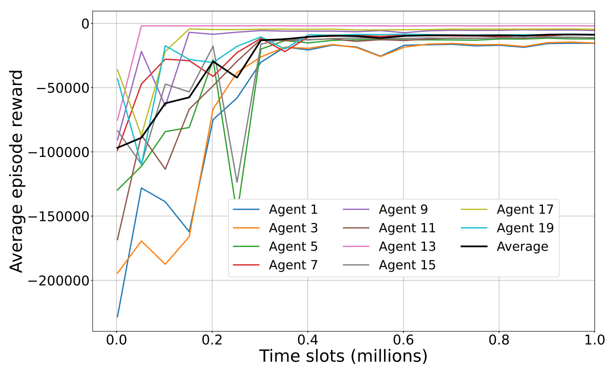


Fig. 13: Rewards of training episodes with separate policies.

reward from the MARL method using a shared policy for the 19-AP 57-device cellular network (Fig. 6a). The blue dashed line represents the average reward, with an exponential moving average curve in orange enhancing clarity. The reward initially improves rapidly, indicating quick learning by the agents. After approximately 750,000 time slots, the reward generally stabilizes, suggesting that each agent has successfully learned an effective and stable policy, resulting in a consistent and favorable cumulative reward.

Fig. 13 displays the rewards for selected agents (only odd-numbered agents for enhanced readability) and average reward of all agents when using separate policies in the same network. During initial training, agents serving devices with low interference (e.g., agents 9 and 13) quickly achieve favorable rewards. Conversely, agents dealing with significant neighbor interference, like agent 5, face early challenges. Despite fluctuations, all agents' rewards generally trend upward, as evidenced by the average reward across all agents. They converge to efficient policies slightly faster than the shared policy approach, achieving convergence within approximately 600,000 time slots. We attribute faster convergence to each agent's ability to learn an individual policy customized for its local topology, interference patterns, and traffic conditions, rather than requiring a single policy to generalize across diverse scenarios experienced by all agents.

E. Model Mismatch

We examined the robustness of the MARL method when trained and tested under different traffic conditions. Performance is considered "unstable" if queue lengths persistently increase over time, "good" if it shows satisfactory QoS compared to the benchmark, and "mixed" if there is a combination of "good" and "unstable" results among the agents.

TABLE V: Training and testing mismatch.

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 V 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. However, policies trained in light traffic show poor performance under medium and heavy traffic conditions.

V. Conclusion

In this paper, we have introduced a novel traffic-driven MARL framework for resource allocation where agents use only locally available information. Numerical results demonstrate that we can achieve packet delay performance comparable to existing genie-aided centralized algorithms. The results also showcase the scalability, robustness, and efficiency of the trained policies across various network sizes and traffic conditions. For future research, it would be interesting to address additional challenges such as user association and beamforming with multiple antennas.

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 their support and useful comments.

References

[1] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, 2011.

[2] 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.

[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] N. Naderializadeh and A. S. Avestimehr, "ITLinQ: A new approach for spectrum sharing in device-to-device communication systems," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1139–1151, 2014.

[7] J. Huang, R. A. Berry, and M. L. Honig, "Distributed interference compensation for wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 5, pp. 1074–1084, 2006.

[8] S. G. Kiani, G. E. Oien, and D. Gesbert, "Maximizing multicell capacity using distributed power allocation and scheduling," in *Proc. IEEE Wireless Communications and Networking Conference*, 2007, pp. 1690–1694.

[9] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.

[10] 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*, 2021, pp. 4720–4724.

[11] 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.

[12] J. Tan, Y.-C. Liang, L. Zhang, and G. Feng, "Deep reinforcement learning for joint channel selection and power control in D2D networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1363–1378, 2020.

[13] Y. S. Nasir and D. Guo, "Deep reinforcement learning for joint spectrum and power allocation in cellular networks," in *IEEE GLOBECOM Workshops*, 2021, pp. 1–6.

[14] Y. Zhang and D. Guo, "Traffic-driven spectrum and power allocation via scalable multi-agent reinforcement learning," in *60th Annual Allerton Conference on Communication, Control, and Computing*, 2024, pp. 1–8.

[15] A. A. Khan and R. S. Adve, "Centralized and distributed deep reinforcement learning methods for downlink sum-rate optimization," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 8410–8426, 2020.

[16] Z. Liu, Y. Chen, Y. Yuan, K. Y. Chan, and X. Guan, "A dynamic power allocation scheme based on multi-agent deep Q-network with environmental awareness for 5G dense cellular networks," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[17] Y. Zhang and D. Guo, "Spectrum allocation and scheduling in conflict graphs via scalable multi-agent reinforcement learning," in *58th Asilomar Conference on Signals, Systems, and Computers*, 2024.

[18] J. Ge, Y.-C. Liang, L. Zhang, R. Long, and S. Sun, "Deep reinforcement learning for distributed dynamic coordinated beamforming in massive MIMO cellular networks," *IEEE Transactions on Wireless Communications*, pp. 4155 – 4169, 2023.

[19] H.-H. Chang, Y. Song, T. T. Doan, and L. Liu, "Federated multi-agent deep reinforcement learning (Fed-MADRL) for dynamic spectrum access," *IEEE Transactions on Wireless Communications*, vol. 22, no. 8, pp. 5337–5348, 2023.

[20] D. Guo, L. Tang, X. Zhang, and Y.-C. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 124–13 138, 2020.

[21] Y. Zhang, "Code implementation of this work," 2024, `https://github.com/yimingzhang2026/MARL-multicell`.

[22] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016, vol. 1.

[23] 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.

[24] 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*, vol. 120, 2020, pp. 256–266.

[25] 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.

Yiming Zhang received his B.E. degree from University of Electronic Science and Technology of China in 2021, his M.S. degree from Northwestern University in 2023. He is pursuing his Ph.D. degree at Electrical and Computer Engineering department of Northwestern University. His current research focus on interference and resource management in heterogeneous networks. His research interests include wireless communications, signal processing and machine learning.

Dongning Guo received his B.Eng. degree from the University of Science and Technology of China, Hefei, his M.Eng. degree from the National University of Singapore, and his Ph.D. degree in Electrical Engineering from Princeton University. He joined the Northwestern University faculty in 2004, where he is currently a Professor in the Department of Electrical and Computer Engineering. He is also a Research Partner with SpectrumX, an NSF Spectrum Innovation Center. He has served as Associate Editor of *IEEE Transactions on Information Theory* and *IEEE Transactions on Wireless Communications*, Editor of *Foundations and Trends in Communications and Information Theory*, and Guest Editor for the *IEEE Journal on Selected Areas in Communications*. He received the NSF CAREER Award in 2007, the IEEE Marconi Prize Paper Award in Wireless Communications in 2010, the Best Paper Award at the IEEE Wireless Communications and Networking Conference in 2017, and the 2023 Bitcoin Research Prize. He also led teams that reached the final matches of the DARPA Spectrum Challenges in both 2014 and 2019. He was elected Fellow of IEEE in 2020. His research interests include wireless networks, blockchain and decentralization, information theory, and machine learning.