Multi-Agent Reinforcement Learning for Wireless User Scheduling: Performance, Scalablility, and Generalization

Kun Yang*, Donghao Li†, Cong Shen*, Jing Yang†, Shu-ping Yeh‡, Jerry Sydir‡
* Department of Electrical and Computer Engineering, University of Virginia, USA
† Department of Electrical Engineering, the Pennsylvania State University, USA
‡ Intel Corporation, USA

2022 56th Asilomar Conference on Signals, Systems, and Computers | 978-1-6654-5906-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/IEEECONF56349.2022.10051992

***Abstract*—We propose a multi-agent reinforcement learning (MARL) solution for the user scheduling problem in cellular networks. Incorporating features of this particular use case, we cast the problem in a decentralized partially observable Markov decision process (Dec-POMDP) framework, and present a detailed design of MARL that allows for fully decentralized execution. The performance of MARL against both centralized RL and an engineering heuristic solution is comprehensively evaluated in a system-level simulation. In particular, MARL achieves almost the same total system reward as centralized RL, while enjoying much better scalability with the number of base stations. The transferability of both MARL and centralized RL to new environments is also investigated, and a simple fine-tuning approach based on a general model trained on a pool of environments is shown to have faster convergence while achieving comparable performance with individually trained RL agents, demonstrating its generalization capability.**

***Index Terms*—User scheduling; Multi-agent reinforcement learning (MARL); Reinforcement learning (RL)**

I. Introduction

Dense deployment of base stations (BS) is an emerging solution to increase the overall throughput of modern cellular networks. The high density requires all base stations to work cooperatively in order to mitigate inter-cell interference with concurrent transmissions. Thus, a well-designed radio resource management (RRM) algorithm is crucial in optimizing the overall throughput of the wireless communication system.

Conventionally, RRM algorithms are largely based on optimization theory, where the design aims at optimizing certain utility that depends on the instantaneous state of the system. This approach is insufficient when the system utility represents the long-term behavior rather than is based entirely on the current state. Additionally, it often involves solving large-scale and non-convex optimization problems, which are computationally challenging.

In recent years, there is a growing interest in applying reinforcement learning (RL) methods to adaptively configure the radio resources to match the deployment environment on the fly. An overview of related works is given in Section I-A. This is a natural "marriage" for several reasons. First, RRM is a closed-loop and sequential operation: configure the resources, observe performance, and fine tune. Second, RRM mostly cares about the *long-term* performance, and its parameters are adjusted at a low rate. Last but not the least, there exist well-established feedback protocols in cellular standards, which provide a built-in mechanism for observing the state and receiving rewards.

Two critical issues that adopting RL for RRM faces are the *scalability* and *generalization*. Dense deployment leads to an exponentially growing action space, which creates significant challenges for training a (deep) RL algorithm. Furthermore, the model training phase of RL does not necessarily have the same environment as the deployment phase, which may be a potentially unseen environment. How to efficiently transfer the trained RL agent to a new environment with fast convergence, is an important research problem.

This paper focuses on a specific RRM use case of *user scheduling* to address the aforementioned challenges. We cast user scheduling as a decentralized partially observable Markov decision process (Dec-POMDP) problem, and then develop a multi-agent reinforcement learning (MARL) framework to make user scheduling decisions. The MARL design takes into account the specific application of cellular user scheduling and centers the design around a deep learning algorithm called Deep Recurrent Q-Network (DRQN) [1], which allows each agent to utilize its own historical information in decision making, thus making it a decentralized executed policy. We demonstrate that the MARL design not only scales much better than its centralized RL counterpart due to its fully decentralized nature, but also achieves almost the same performance. Furthermore, we investigate the transferability of both centralized RL and MARL with respect to mismatched or even unseen environments (between training and testing), and empirically evaluate a transfer learning method that first trains a general RL agent based on a pool of environments, and then fine-tunes when deployed. The advantage of faster convergence is observed for both centralized RL and MARL, although the gain of the latter is smaller than the former.

A. Related Works

There have been a growing body of literature that apply RL to solving wireless network optimization problems, such as spectrum access and sharing [2]–[4], mobility management [5]–[7], and power control [8], [9]. Limiting to the scope of

The first two authors contributed equally to this work.

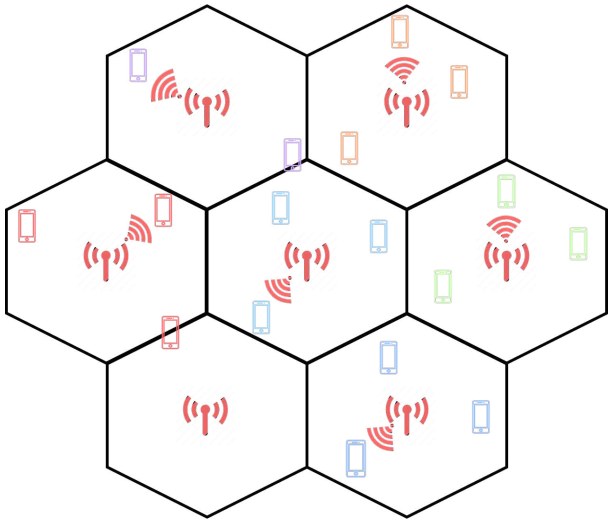

Fig. 1: Illustration of the system model. In this example, BSs are regularly positioned with a conventional hexagonal structure, while UEs are randomly dropped. Each BS can choose to serve one UE (indicated by the airwave symbol), or turn off (no airwave symbol).

this work, RL has been used for user selection in [10]. More recently, [11] uses deep RL to solve radio resource allocation problems in massive multi-input multi-output (MIMO) systems. [11]–[13] are the closer literature to our work, using either deep RL or MARL to control downlink power and user selection. More specifically, [11] uses centralized deep RL algorithms to control transmit power as well as subchannel allocation in a multi-cell system. Similarly, [12] and [13] use MARL to control the downlink power and select users with the exchange of neighbor observations to maximize the overall throughput of the wireless system.

B. Paper Organization

The remainder of this paper is organized as follows. The system model and the Dec-POMDP formulation are described in Section II. Details of both MARL and centralized RL designs are given in Section III. Experiment results are reported in Section IV. Finally, Section V concludes the paper.

II. System Model

We first present the communication model for a cellular user scheduling problem, and then describe how to cast it in the framework of decentralized partially observable Markov decision process (Dec-POMDP) with networked agents.

A. Communication Model

A multi-cell wireless network is considered, as depicted in Fig. 1. There are N_a base stations (BS) and N_u user equipment devices (UE) located inside an area with a size of $d_0 \times d_0$. We consider a time slotted system. The UEs, after being randomly dropped in the area, take random walks following

$$[x(t), y(t)] = [x(t-1), y(t-1)] + v_0 \left[\delta_x, \delta_y\right],$$

where (δ_x, δ_y) follow independently uniform distribution on $[-1, 1]$, and v_0 thus represents the maximum step size a UE can move in one time slot. Boundary conditions are handled with mirror-back.

We adopt a standard 3GPP pathloss model with a specified antenna pattern [14]. Specifically, we assume each BS has a maximum transmit power of 10 dBm, and the pathloss between BS i and UE j is characterized by

$$PL_{i,j} = 15.3 + 37.6 \log(d_{ij}) + L_{ow}, \quad d > d_0$$

where $L_{ow} = 10$ dB and d_{ij} is the distance between UE j and BS i. We set $d_0 = 1$m and re-drop (or bounce) the UE if it enters this area around the BS. We also adopt a parabolic antenna pattern, defined as

$$A(\theta) = \min\left\{12\left(\frac{\theta}{\theta_{3\text{dB}}}\right)^2, A_m\right\}$$

with $\theta_{3\text{dB}} = \pi/3$ and $A_m = 20$ dB. $A(\theta)$ is then subtracted from the transmit power when computing the received power at direction θ. To simplify the problem, we only consider pathloss in the user scheduling problem, which is suitable when the timescale of the scheduling decision is large. This simplification also allows for a (relatively) less random wireless environment, where the dynamics mainly come from the geometry and user movement.

We assume that all BSs and UEs are on the same frequency, and each BS serves at most one UE at a time. In each time slot, if UE j is served by BS i, its data rate at time t is given by

$$C_{i,j}(t) = \log\left(1 + \mathsf{SINR}_{i,j}(t)\right)$$

where $\mathsf{SINR}_{i,j}(t)$ denotes the signal-to-interference-plus-noise ratio (SINR) that captures both the background noise and all interferences from non-serving BS's $\{k \in [N_a] : k \neq i\}$. The overall system utility, however, often cannot simply consider the *instantaneous* data rate for various reasons (e.g., fairness). We thus adopt the *long-term average rate* for UE j as [13]:

$$r_j(t) = \alpha r_j(t-1) + (1-\alpha) C_{i,j}(t), \tag{1}$$

if UE j is served by BS i at time t. If no BS is serving this UE at t, the second term is 0. $\alpha \in [0, 1]$ is the discount factor.

B. The POMDP Formulation

A Dec-POMDP can be written as

$$\mathcal{M} = (\mathcal{S}, \{\mathcal{A}_i\}_{i\in[N_a]}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \{O\}_{i\in[N_a]}, N_a, \gamma)$$

for $[N_a]$ agents. In this formulation, each agent i has its local observation $o_i \in \mathcal{Z}$ which is sampled from a global state $s \in \mathcal{S}$ according to $o_i = O_i(s)$. Agent i generally has a different observation o_i from other agents and can thus take different action $a_i \in A_i$. The joint action is denoted as $a_t = (a_{1,t}, a_{2,t}, \cdots, a_{N_a,t})$. When a joint actions u is taken by the agents under a given state s, the state transit probability $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, written as $P[s' \mid s, u]$, denotes the probability of transitioning to next state s', and a global

reward R is collected. Our goal is to maximize the discounted accumulated reward:

$$G(t) := \mathbb{E}\left[\sum_{\tau=0}^{+\infty} \gamma^{\tau} R(t+\tau)\right] \tag{2}$$

by allowing each agent to choose which UE (if any) to serve at each time slot t.

Given this formulation, we now describe how to model the user scheduling problem as a Dec-POMDP. We consider each BS as an agent, and present the key elements as follows.

Agents. Each BS is an agent in the MARL formulation.

State. The global state of the entire system consists of the relevant UE information, i.e., position and moving average data rate in Eqn. (1). We note that the discount factor α helps balance the services to all UEs, including the ones that experience unfavorable propagation, by decaying the moving average data rate of UEs that have not been served.

Observation. Each agent is only allowed to observe the nearest K UEs. The local observation of each agent contains the UE index, its relative position to the agent (BS), and the moving averaged data rates of the UE.

Action. In this work, we consider a simple action space where each agent can choose to serve one of the K nearest UEs with a maximum transmit power, or not serve any UE by turning off (to reduce its interference to other UEs served by nearby BSs). We also comment that for the algorithms presented later in this paper, if more than one agent chooses the same UE to serve, only the agent corresponding to the strongest received power at the UE successfully serves this UE, while other agents are counted as interference. This is also a relatively pessimistic option, as advanced mechanisms such as Coordinated Multipoint (CoMP) [15] may allow multiple BSs to simultaneously serve the same UE and thus increase the system utility.

Reward. The global reward is defined as the sum of all moving averaged data rates, truncated at a threshold value to reflect the maximum rate limitation in practical systems. More specifically, the overall system reward can be written as

$$R(t) = \sum_{j \in [N_u]} \max\{r_j(t), r_{\text{TH}}\}$$

where r_{TH} is the UE rate threshold.

III. Algorithm Designs

Maximizing the objective in (2) falls into the category of reinforcement learning (RL), and we propose a multi-agent reinforcement learning (MARL) design that is tailored for the specific wireless user scheduling problem. In addition, we also present a centralized RL design that serves two purposes: (1) It serves as a benchmark for MARL, to measure how much "coordination" among clients (by allowing a central decision making based on the complete system state) can benefit the policy; (2) Its performance against a baseline design, which will be elaborated in the simulation section, characterizes the potential gain RL-based designs may have.

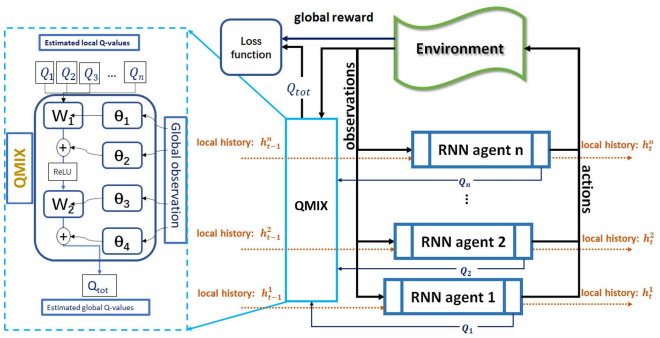

Fig. 2: The MARL design for user scheduling.

A. MARL Design

Our proposed MARL design is based on two well-known methods: DRQN [1] and QMIX [16], but with substantial changes that reflect the "domain knowledge" of our wireless problem. An overall illustration of the proposed MARL design is given in Fig. 2. The design involves a Deep Recurrent Q-Network (DRQN) model for agents to estimate their local Q functions, and a QMIX network to estimate the global Q function from the local Q functions. More specifically,

- **DRQN.** DRQN operates as follows

$$\tau_0^{(i)} = 0; \tau_t^{(i)} = h(o_t^{(i)}, \tau_{t-1}^{(i)}; \theta_h),$$
$$Q_{\text{loc}}^{(i)} = \max_a Q^{(i)}(\tau_t^{(i)}, a; \theta_q).$$

  Basically, it utilizes the history information in $\tau_{t-1}^{(i)}$ which is generated from network h defined by θ_h. We note that this network only requires *local* history information, i.e., it does not require any information from other agents. This feature is critical in achieving better *scalability*, as the increase of agents does not scale the computation or storage requirement of each agent much. At the same time, another network $Q^{(i)}$ (defined by θ_q) generates the local Q value with local information $\tau_t^{(i)}$. The DRQN network incorporates both functions h and $Q^{(i)}$.
- **QMIX.** Local Q functions alone are insufficient to realize cooperation in an MARL setting. We thus adopt the method of QMIX [16] and construct the global Q function as another function of local Q functions:

$$Q_{\text{tot}} = f_{\text{QMIX}}(Q_{\text{loc}}; \theta_{\text{tot}}(o_t; \theta_p))$$

  where Q_{loc} is a vector of local $Q_{\text{loc}}^{(i)}$. As long as $\frac{\partial Q_{\text{tot}}}{\partial Q_{\text{loc}}^{j}} > 0$, local Q functions will produce the same choice of actions with the global Q function [16]. With QMIX, our network can be trained with back propagation by minimizing

$$E_{\{s,a,s'\}}[(Q_{\text{tot}}(s,a) - (r(s,a,s') + \gamma \arg\max_{a'} Q_{\text{tot}}(s',a')))^2].$$

Finally, the action taken by agent i at time t is

$$a_{i,t} = \arg\max_a Q^{(i)}(\tau_t^{(i)}, a; \theta_q). \tag{3}$$

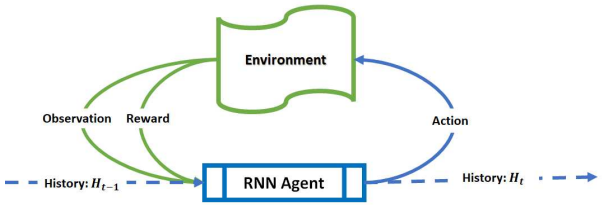

Fig. 3: The centralized RL design for user scheduling.

B. Centralized RL Design

Based on the MARL design in Section III-A, we further develop a centralized RL method for the user scheduling problem. Since the goal for the centralized RL is to facilitate our investigation of MARL, we reuse DRQN in order to have a fair comparison. The action space of centralized RL is the combination of actions from all BSs, and the basic training steps remain the same as discussed in the previous MARL section. Compared with the MARL design, the centralized RL is "simple" in the sense that the RNN network takes the global observation $o_t = [o_t^1, o_t^2, \cdots, o_t^N]$ as the input and directly outputs the global Q-function Q_t. Fig. 3 illustrates the centralized RL design for user scheduling, and the action Centralized RL agent takes as time t is

$$a_t = \arg\max_a Q(\tau_t, a; \theta_q).$$

IV. Experiment Results

A. Simulation Setup

We evaluate the performance of the proposed MARL design in a system-level simulation. In the simulator, the environment has N stationary BSs that are randomly distributed within a bounded area with a minimum distance of d separating them. The simulator will randomly generate the initial positions for the M UEs, and these UEs will randomly move. The detailed setting of the environment follows the description in Section II. All the results shown in this section are averaged of 10 randomly generated environments unless otherwise specified. The comparison of different algorithms is fair because they share the same set of multiple random seeds in generating the environment traces in the simulation.

We compared our **MARL** method with two baselines. One is a rule-based greedy algorithm, where each BS greedily serves the user with the best channel quality until the long-term local data rate reaches the predefined threshold. The BS then serves the second best UE. We denote this simple greedy algorithm as the **Heuristic** algorithm. The other baseline we would like to compare our MARL design with is the **Centralized RL**, described in Section III-B. This is basically a centralized version of DRQN, where the DRQN network takes the global observation as the input and provides actions for all agents.

B. System Reward and Training Efficiency

We compare the training efficiency between Centralized RL, MARL, and Heuristic mentioned above. In our design of the centralized RL agent, the size of action space grows exponentially with the number of BSs, rendering the training difficult. We thus set the number of BSs to 4, to allow for a fair comparison of training efficiency between RL and MARL. The scalability of these two methods will be investigated in Section IV-C.

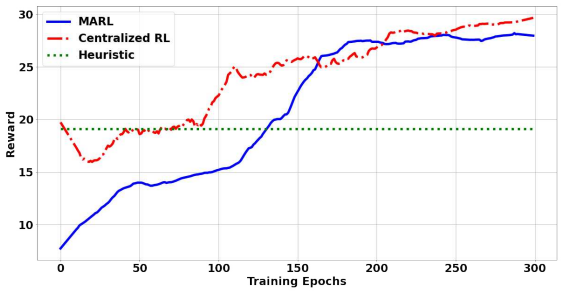

Fig. 4: Average training results from a 4-BS 10-UE environments.

Method	Epochs to optimal	Total Reward
Heuristic	N/A	19.10 +/-1.08
Centralized RL	200	24.31 +/- 1.09
MARL	200	23.65 +/- 0.98

TABLE I: Average training results on 10 different environments.

From Fig. 4 and Table I, we can see that both methods can outperform the rule-based heuristic method with roughly 150 epochs of training[1], and Centralized RL converges a little faster than MARL. After 300 training epochs, both methods can outperform Heuristic by approximately 38% and almost converge to the same operating point. On the other hand, because the centralized RL agent has a complete view of the system (i.e., global observation), it has a better starting point than MARL. This leads to better early performance of Centralized RL over MARL, which is observed in the results.

C. Scalability

The "curse of dimensionality" prevents the centralized RL to scale with the number of agents, as the action space grows exponentially and convergence cannot be guaranteed anymore. This scalability issue associated with centralized RL is particularly harmful when *deep learning* is adopted. First, a large action space typically leads to a large output layer. Second, the exponentially expanding action space means adequate exploration quickly becomes difficult if not infeasible.

These two issues are captured in the simulation results reported in Fig. 5, where we repeat the experiment setting described in Section IV-A with different number of agents. We report the *total* system rewards, which is why they generally increase with the number of agents. We can see that when the number of agents is small, i.e., between 4 and 6 in our particular user scheduling simulation, we have the expected

[1]Our training epoch is similar to the training episode defined in Algorithm 3 of [17]. We keep a replay buffer and sample a mini-batch of data points for training at each time slot. Meanwhile, we also use our current agent to sample a new trajectory and feed it into the replay buffer.

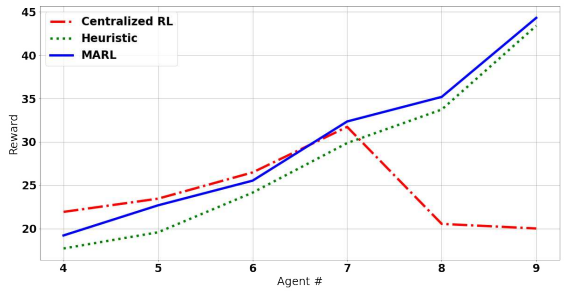

Fig. 5: Total rewards versus the number of agents.

behavior that Centralized RL has a slight advantage over MARL, both of which outperform Heuristic. This is consistent with the results in Section IV-B. However, when the number of agents continues to increase, we can see that Centralized RL first falls below MARL, and then its convergence is severely impacted such that the total rewards decrease with more agents in the system. We should emphasize that these results are obtained by training for the same amount of epochs across different number of agents. Conceivably, the performances can be improved by increasing the training epochs, but it still highlights the scalability limitation of Centralized RL. On the other hand, the centralized-training decentralized-execution MARL solution scales well with the number of agents, and has a much better performance. Nevertheless, its advantage over Heuristic also becomes smaller at 8 or 9 agents, mainly because the centralized training also becomes more difficult.

D. Transferability

1) The Problem: One issue of the experimental results so far is that both training of the RL agents and testing are carried out in the same environment. In other words, there is no statistical mismatch between training and testing. In reality, however, this is often not the case. What is more likely is that data collected to train the RL agent may not fully represent the actual deployment environment. It is of interest to know how RL agents trained in one environment behave in others, and how one can improve the generalization capability of MARL for wireless user scheduling.

We first take a look at how rewards are affected by the different deployments. To see this, we have randomly generated 100 environments, and run the Heuristic algorithm (for its computational efficiency) to get the system rewards for these deployments. The deployment setting follows that in Section IV-A, with 4 fixed BSs and 10 randomly dropped UEs. Fig. 6 plots the histogram of system rewards. The results show that the system reward can change drastically under the same algorithm, even in such a setting with limited randomness.

A natural follow-up question from Fig. 6 is that, if we take MARL agents trained in one environment and then directly apply them to a different one, how much of a performance degradation, if any, we will observe? To answer this question empirically, we randomly generated 10 environments (with

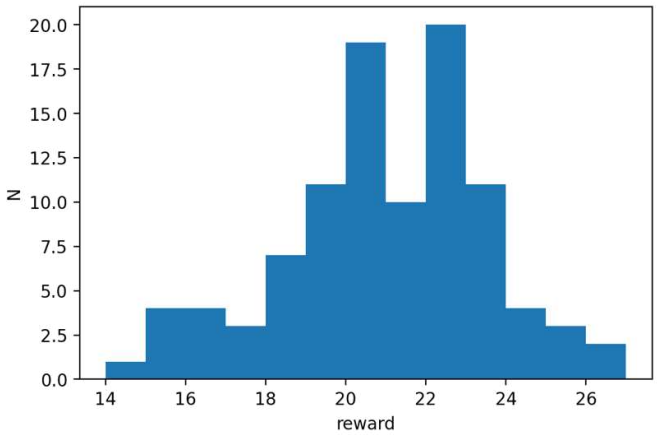

Fig. 6: Histogram of the system rewards with Heuristic across 100 randomly generated environments.

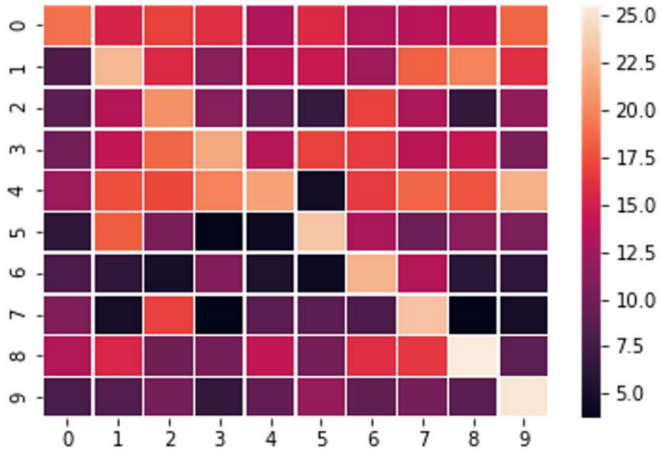

Fig. 7: Rewards of the agents trained in one environment and tested in another.

the same procedure described previously) and then train the MARL agents for each deployment. For every trained set of agents, we then test their performances over the other 9 deployments, and plot the resulting system rewards in Fig. 7. In this figure, the row indices stand for the training environments, while the indices of the columns are for the testing environments. As we can see from this figure, there are a few environment pairs where the performances do transfer well, but by and large, the transferability is very limited.

2) The Solution: This limitation essentially requires one to train the RL model from scratch for every new environment, which is undesirable. Alternatively, a strategy to reduce the computational cost is to train a general model that performs relatively well across a large number of environments, and then fine-tune the general model on the specific environment at the deployment time.

To evaluate this approach, we have reused the same 10 environments from those in Fig. 6, and trained the general model according to the following steps.

1) Initialize the common model.
2) For each outer loop, randomly sample an environment.

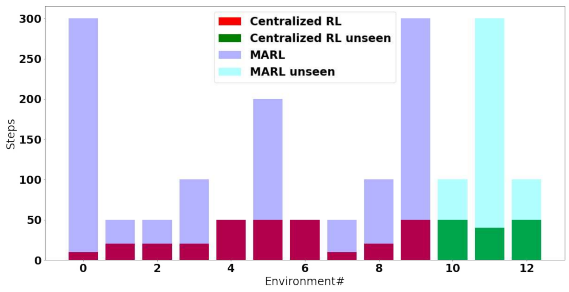

Fig. 8: The required fine-tuning epochs from the general model to achieve 90% performances of individually trained RL agents. General model is trained on environments 0 to 9, and fine-tuned on all 13 environments.

3) Train the RL model for n epochs. Test on all 10 environments and take the average.
4) Repeat steps 2) and 3) until a good result is achieved.

After these steps, we then fine-tune the general model on the specific environment in which we deploy the RL agents, using the same training mechanism as described previously. Our hope is that with a general model that performs reasonably well across a large number of environments, it presents a good initialization model to have quicker convergence than starting from scratch (i.e., a random initial model). The experimental results are reported in Fig. 8. In this figure, we first pool the 10 environments together to train a general model with 1000 epochs, and then fine-tune the model for each of the 10 environments. Fig. 8 reports the number of additional steps (epochs) fine-tuning requires to reach at least 90% of the reward that is achieved by Centralized RL, which is individually trained for this environment. We can see that for fine-tuned Centralized RL, we generally require fewer than 50 training steps of fine-tuning. This is a significant advantage over training from scratch, which often requires 300 or more steps to converge. Fine-tuned MARL, on the other hand, has similar but smaller advantage for most environments, mainly because the fine-tuning of MARL is more difficult than Centralized RL. This is also why for some environments (e.g., 0 and 9), MARL does not have an advantage compared with training from scratch. Nevertheless, we have an average saving of over 80% of the training resources for fine-tuning MARL across all 10 random environments. An interesting future research direction is how to improve fine-tuning MARL, and we believe the difficulty is mainly from the QMIX, since different environments often have different QMIXs and tuning from the general QMIX to local QMIXs is difficult.

As a final result, we have also fine-tuned the general model, which was obtained by training on environments 0 to 9, for three unseen environments (indexed as 10, 11, and 12). We can see that the generalization capability of such general model with fine-tuning also carries over to unseen environments, where we again need only 50 steps for Centralized RL and 100 steps for most of the MARL (with one exception).

V. Conclusion and Discussion

We have developed a multi-agent reinforcement learning (MARL) solution for user scheduling. This method enjoys comparable performance to the centralized RL, both of which outperform the simple engineering heuristic solution. More importantly, we have seen that our MARL design allows for better scalability (with the number of base stations) in the system than centralized RL, which is an important feature that is highly desirable in practice. On the other hand, transferability of both MARL and centralized RL was empirically studied, and a simple fine-tuning approach based on a general model trained on a pool of environments was shown to achieve faster convergence while achieving comparable performance with individually trained RL agents.

References

[1] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *2015 AAAI Fall Symposium Series*, 2015.
[2] Z. Wang, Z. Ying, and C. Shen, "Opportunistic spectrum access via good arm identification," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov 2018, pp. 673–677.
[3] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, June 2018.
[4] C. Gan, R. Zhou, J. Yang, and C. Shen, "Cost-aware learning and optimization for opportunistic spectrum access," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 1, pp. 15–27, March 2019.
[5] Z. Wang, L. Li, Y. Xu, H. Tian, and S. Cui, "Handover control in wireless systems via asynchronous multiuser deep reinforcement learning," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4296–4307, Dec 2018.
[6] Y. Zhou, C. Shen, and M. van der Schaar, "A non-stationary online learning approach to mobility management," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 1434–1446, Feb 2019.
[7] C. Wang, R. Zhou, J. Yang, and C. Shen, "A cascading bandit approach to efficient mobility management in ultra-dense networks," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019, (Invited Paper).
[8] C. Shen, R. Zhou, C. Tekin, and M. van der Schaar, "Generalized global bandit and its application in cellular coverage optimization," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 218–232, Feb 2018.
[9] J. Cui, Y. Liu, and A. Nallanathan, "Multi-agent reinforcement learning based resource allocation for UAV networks," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2019, early access.
[10] E. Ghadimi, F. D. Calabrese, G. Peters, and P. Soldati, "A reinforcement learning approach to power control and rate adaptation in cellular networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–7.
[11] X. Guo, Z. Li, P. Liu, R. Yan, Y. Han, X. Hei, and G. Zhong, "A novel user selection massive mimo scheduling algorithm via real time DDPG," in *GLOBECOM 2020*. IEEE, 2020, pp. 1–6.
[12] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE J. Select. Areas Commun.*, vol. 37, no. 10, pp. 2239–2250, 2019.
[13] N. Naderializadeh, J. J. Sydir, M. Simsek, and H. Nikopour, "Resource management in wireless networks via multi-agent deep reinforcement learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3507–3523, 2021.
[14] 3GPP, "Simulation assumptions and parameters for FDD HeNB RF requirements," Tech. Rep. R4-092042.
[15] P. Marsch and G. P. Fettweis, *Coordinated multi-point in mobile communications: from theory to practice*. Cambridge University Press, 2011.
[16] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.
[17] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv preprint arXiv:1712.01275*, 2017.