Distributed Reinforcement Learning for Low-delay Uplink User Scheduling in Multicell Networks

Apostolos Destounis, Dimitrios Tsilimantos
Huawei Technologies, Paris Research Center
{apostolos.destounis, dimitrios.tsilimantos}@huawei.com

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001034

***Abstract*—In this paper we investigate the problem of uplink scheduling in a multicell system in order to minimize the total queuing delay at the mobile devices. The proposed setting introduces an environment with multiple interacting decision makers: Base Stations are considered as agents who have partial view of the system and interact with each other through interference generated by their scheduled devices for uplink transmissions. In addition, since traffic and channel processes are unknown, finding the optimal global scheduling policy can be modeled as a multi-agent reinforcement learning problem. In this work, we propose distributed learning algorithms based on policy gradient to tackle this problem and investigate the impact of information exchange between Base Stations. Our results illustrate that the proposed algorithm outperforms standard schedulers, such as Proportional Fair and MaxWeight and that information exchange is crucial for challenging problem instances, such as topologies with many devices on the cell edge and/or high traffic demands.**

***Index Terms*—Scheduling, Multi-Agent Reinforcement Learning**

I. Introduction

In the forthcoming next generation of wireless networks, a large number of devices is expected to be connected to every Base Station (BS). Since a BS can normally observe the conditions (e.g. channel and traffic states, delays) of only its associated devices, radio resource allocation decisions need to be taken in a distributed fashion, based only on this local information. In this paper, we study the problem of low-delay uplink scheduling, where the decision of one BS has an impact on other BSs due to the interference from its scheduled device.

A simple approach to handle this problem is to use standard schedulers at each BS, such as Proportional Fair [1] or MaxWeight [2], ignoring however the impact of interference. On the other hand, BS coordination for resource allocation decisions can bring significant gains in performance. For example, devices with similar traffic and channel characteristics are clustered into a pre-specified number of classes in [3] and coarse information for each class is exchanged among BSs. A scheduling algorithm for the LTE uplink is proposed in [4], where BSs are partitioned in clusters and BSs in a cluster exchange information about their devices in order to achieve proportional fairness in the system. Taking the traffic of the devices into account, a weighted sum-rate optimization problem at each scheduling slot is solved at [5], where the weights of each device correspond to its traffic state. However, the per-slot optimization requires a relatively large amount of signalling among BSs and does not necessarily lead to long term delay minimization.

Motivated by the success of deep Reinforcement Learning (RL), there has been considerable recent work dealing with Multi-Agent RL (MARL) for distributed radio resource management in cellular networks [6]. For example, a deep Q-learning algorithm is proposed in [7], where the BSs take distributed decisions based on information of their own devices and neighboring BSs. The use of various MARL algorithms for a similar setting is studied in [8], where Multi-Agent Deep Deterministic Policy Gradient (MADDPG) is empirically shown to have the best performance. Distributed power control based on Proximal Policy Optimization (PPO) is presented in [9] with identical neural networks at each BS. MARL has also been applied in more complex settings, such as heterogeneous networks in [10] and networks with multiple BS antennas in [11], where a critic with a full view of the system is used in both works. In all [7]-[11] the training is centralized with the objective to maximize the system sum-rate. Finally, the authors in [12] use MARL for scheduling and power control policies to achieve proportional fairness among the service rates of the devices. Their approach is based on double deep Q-learning with centralized training and same policy for each BS.

In contrast to the above works, which focus on the physical layer, we take into account the fact that devices in a wireless network have traffic coming from higher layers, which has to be served with low delay. Distributed radio resource allocation for such settings has been investigated in [13], where downlink power control is used in a multicell system to minimize the queue length at the devices. However, traffic and channel statistics are assumed known and the control problem is solved offline, unlike our work where the scheduling policy is learned in a distributed way at each BS. The latter is desirable in practice as no data exchange to a central controller is required.

Regarding uplink transmissions, there are also works on deep MARL for distributed multiple access with devices as agents that contend to access the channel. An independent Q-learning algorithm to maximize system throughput, while co-existing with devices that run ALOHA and TDMA protocols, is presented in [14], and a framework to dynamically select the CSMA window is proposed in [15]. Unlike our approach, these works are usually limited to a single cell with collisions and cannot easily exploit communication among agents.

In summary, the contributions of our work are as follows:

- We formulate the problem of distributed low delay uplink scheduling as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP).

- We present algorithms based on policy gradient that enable distributed learning with (i) very low information overhead and (ii) convergence guarantees to a distributed scheduling policy that is a local optimal point.
- We illustrate through extensive simulations that the proposed RL algorithms outperform standard schedulers and investigate the impact of additional information exchange.

II. System Model

We consider the uplink of a network with N BSs and K devices, as illustrated in Fig.1, where each device is associated to a single BS. Specifically, a device k is associated to BS $b(k)$ and the set of devices served by BS n is $\mathcal{K}(n)$. Time is slotted with a slot duration of T_s and at each slot t, A_t^k bits of information arrive for transmission at device k according to a random process with $\mathbb{E}\left[A_t^k\right] = \lambda_k$. Each device has a queue of length x_t^k, measured in bits, where unserved traffic waits for transmission. We assume a block fading channel model, where the channel gain between device k and BS n at slot t is denoted as $g_t^{k,n}$. If at slot t each BS n schedules a device $u_t^n \in \mathcal{K}(n)$ for transmission, then this device transmits

$$R_t^{u_t^n} = WT_s \log_2\left(1 + \frac{g_t^{u_t^n,n} P}{1 + \sum_{j \neq n} g_t^{u_t^j,n} P}\right) \quad (1)$$

bits in the slot, where P is the uplink transmission power, normalized by the receiver noise power at the BS, and W is the bandwidth. The transmission rate of not scheduled devices is zero, i.e. $R_t^k = 0, \forall k \neq u_t^n, \forall n$. The queues then evolve as

$$x_{t+1}^k = \left[x_t^k - R_t^k\right]^+ + A_t^k, \forall t \geq 0, \forall k \in \mathcal{K}. \quad (2)$$

At the beginning of slot t, each BS n can directly observe the queue and the channel states of its associated devices, which we denote as its local state $\mathbf{s}_t^n \triangleq \{(x_t^k, g_t^{k,n})\}_{k \in \mathcal{K}(n)}$. In addition, BSs may exchange their local states over a control interface. Therefore, a BS n exchanges state information with BSs $\mathcal{N}(n)$ and its observation at the beginning of slot t is $\mathbf{o}_t^n = \{\mathbf{s}_t^i\}_{i \in \mathcal{N}(n) \cup \{n\}}$. A policy is then defined as follows:

Definition 1: A scheduling policy π^n is a function which maps an observation $\mathbf{o}^n$ of BS n to a probability distribution over $\mathcal{K}(n) \cup \emptyset$.

The empty set $\emptyset$ denotes the decision to not schedule a device. Thus, the system of queue lengths evolves under a joint policy π, which is the product of the individual policies of each BS,

$$\pi(\mathbf{u}_t | \mathbf{x}_t, \mathbf{g}_t) = \prod_{n=1}^{N} \pi^n(u_t^n | \mathbf{o}_t^n). \quad (3)$$

Since each device has a traffic to be served, the goal of the network is to find a policy for taking actions at the beginning of each slot in order to keep the latency at the devices small. A suitable objective towards this goal is to minimize the average total weighted queue length, which is equal to the average delay if $w^k = 1/\lambda_k, \forall k$. For a discount factor $\gamma < 1$, we get

$$\min_{\pi^1, \ldots, \pi^N} J(\pi) \triangleq \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \sum_{k=1}^{K} w^k x_t^k\right]. \quad (4)$$

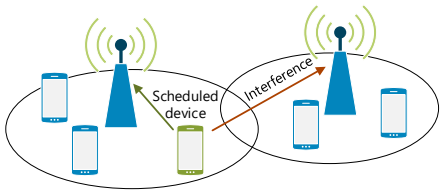


Fig. 1: Example of system model; the scheduled device causes interference to nearby BSs

Similar to the single agent fully observable setting, we can define the local value $V_\pi^n(\mathbf{o}^n)$ and advantage $A_\pi^n(\mathbf{o}^n, u^n)$ functions of each BS under joint policy π as follows:

Definition 2: The local value and advantage functions of BS n under policy π are defined as

$$V_\pi^n(\mathbf{o}^n) = \mathbb{E}_{\mathbf{u}_t \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{k=1}^{K} w^k x_t^k \middle| \mathbf{o}_0^n = \mathbf{o}^n\right],$$

$$A_\pi^n(\mathbf{o}^n, u^n) =$$

$$\mathbb{E}_{\mathbf{u}_{t>0} \sim \pi}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{k=1}^{K} w^k x_t^k - V_\pi^n(\mathbf{o}^n) \middle| \mathbf{o}_0^n = \mathbf{o}^n, u_0^n = u^n\right].$$

The expectation is over the distribution of states and scheduling of other BSs, given the local observation and scheduling decision of BS n. Since each BS can schedule only one of its own devices based on local information, the setting corresponds to a Dec-POMDP [16], where each BS is an agent. As channel and traffic statistics are unknown, MARL is a fitting tool to find a scheduling policy in an online manner.

III. MARL Algorithms for Device Scheduling

Due to the high dimensional state space of the observations, we consider the scheduling policy of BS n as a parametric function. More specifically, we denote as $\pi^n(u^n | \mathbf{o}^n; \boldsymbol{\theta}^n)$ the probability that BS n schedules device u^n under observation $\mathbf{o}^n$ and given parameters $\boldsymbol{\theta}^n$ that depend on the function form. They can be for instance weights and biases of a neural network or weights of a policy that is parameterized as a linear combination of features.

A. Learning a Distributed Schedule via Policy Gradient

In this work, we will use a policy gradient approach, where the objective is to update the parameters of each agent so that they converge to policies with a small cost $J(\pi)$. The motivation of using such an approach comes from the following result, adapted from [17]:

Theorem 1: Define $C_t = \sum_{t' \geq t} \gamma^{t'-t} \sum_{k=1}^{K} w^k x_{t'}^k$ the cost-to-go and $b^n(\mathbf{o}^n)$ any function of the local observation. Then,

$$\nabla_\theta J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\sum_{n=1}^{N} \sum_{t=0}^{\infty} (C_t - b^n(\mathbf{o}_t^n)) \nabla_{\theta^n} \log \pi^n(u_t^n | \mathbf{o}_t^n; \boldsymbol{\theta}^n)\right],$$

where the expectation is over the trajectories that result from following the joint policy with parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^N]$. The above implies that if the total cost-to-go C_t is known to all BSs at each slot t, each BS can estimate the block

of the gradients for the joint policy that corresponds to its own parameters $\boldsymbol{\theta}^n$. Thus, the joint scheduling policy applied across all BSs will be updated in a direction whose expectation is the gradient of the total cost $\nabla_\theta J(\boldsymbol{\theta})$ in a totally distributed manner. In practice, the system is run in episodes and C_t can be observed at the end of each episode from the instantaneous costs. From an implementation perspective, after running M episodes of length T slots with the current joint policy (without loss of generality we assume fixed episode length), the quantity

$$\mathbf{d}^n = \frac{1}{MT}\sum_{m=1}^{M}\sum_{t=0}^{T-1}(C_t - b^n(\mathbf{o}_t^n))\nabla_{\theta^n}\log\pi^n(u_t^n|\mathbf{o}_t^n;\boldsymbol{\theta}^n) \quad (5)$$

is an approximation of the gradient of each BS with large enough horizon T for a small bias and a high number M of episodes for reduced variance of the gradient estimator. The function $b^n(\mathbf{o}^n)$ is used in order to further reduce the variance of the estimator in (5), similar to the use of a baseline in single-agent RL with full state observation (see e.g. [18, Ch. 13.4]). In this work, we model it as a function $V^n_{\boldsymbol{\phi}^n}$ parametrized by $\boldsymbol{\phi}^n$, which for a given joint policy tries to approximate the local value function of Definition 2. In fully observable single agent settings, this baseline is known to minimize the variance [18, Ch. 13.4]. The update of $\boldsymbol{\phi}^n$ at the end of M episodes that follow the joint policy is then performed as

$$\boldsymbol{\phi}^n = \arg\min_{\phi'}\frac{1}{MT}\sum_{m=1}^{M}\sum_{t=0}^{T-1}\left(V^n_{\phi'}(\mathbf{o}_t^n) - C_t\right)^2. \quad (6)$$

A first way to learn a distributed uplink scheduling policy is that each BS runs a simple policy gradient algorithm [18, Ch. 13.3] with a gradient descent of step size a for its policy parameters $\boldsymbol{\theta}^n$ using (5) and then updates its baseline parameters $\boldsymbol{\phi}^n$ by (6). Theorem 1 suggests that *the joint policy is updated, in expectation, in the direction of the gradient* $\nabla_\theta J(\boldsymbol{\theta})$, therefore statistically reducing the total cost. With the choice of b^n as the local value function of BS n, the term $C_t - b^n(\mathbf{o}_t^n)$ in (5) is in fact the local advantage function of BS n under the current (joint) policy π. This observation can help to improve the sample efficiency of the estimator of the gradient. To this end, we use the Generalized Advantage Estimation (GAE) technique [19] to calculate at each slot t the estimation $\hat{A}_t^n$ of the advantage function by

$$\hat{A}_t^n = \sum_{\tau=t}^{T-1}(\gamma\beta)^{\tau-t}\left(\gamma V^n_{\phi^n}(\mathbf{o}^n_{\tau+1}) - \sum_{k=1}^{K} w^k x^k_\tau - V^n_{\phi^n}(\mathbf{o}^n_\tau)\right), \quad (7)$$

where $\beta < 1$ is a GAE parameter. The learning process is presented in Alg. 1. Despite its good theoretical properties, applying a simple policy gradient has practical issues, such as high variance and slow convergence speed, which motivate us to adapt more advanced methods proposed in the RL literature.

Algorithm 1 Global Cost Vanilla Policy Gradient (VPG-GC)

Input: Learning rate a, GAE parameter β, initialize $\boldsymbol{\theta}^n, \boldsymbol{\phi}^n$

1: **for** $\ell = 0, 1, 2, \ldots$ **do**:
2: **for** $m = 1, 2, \ldots, M$ **do**:
3: Run policy with $\boldsymbol{\theta}^n$ and store the trajectory of local observations $\left\{(\mathbf{o}^n_{m,t}, u^n_{m,t}, C_{m,t})\right\}_{t=0,\ldots,T-1}$
4: Calculate $\{\hat{A}^n_{m,t}\}_{t=0,\ldots,T-1}$ according to (7)
5: $\mathbf{d}^n \leftarrow \frac{1}{MT}\sum_m\sum_t \hat{A}^n_{m,t}\nabla_{\theta^n}\log\pi^n(u_t^n|\mathbf{o}_t^n;\boldsymbol{\theta}^n)$
6: $\boldsymbol{\theta}^n \leftarrow \boldsymbol{\theta}^n - a\mathbf{d}^n$ (or via Adam algorithm)
7: $\boldsymbol{\phi}^n \leftarrow \arg\min_{\phi'}\frac{1}{MT}\sum_m\sum_t\left(V^n_{\phi'}(\mathbf{o}_t^n) - C_t\right)^2$

B. Enhancing Policy Gradient - Proximal Policy Optimization

In order to overcome the above issues, Trust Region Policy Optimization (TRPO) has been proposed in [20] for a fully observable single agent RL setting. Its main ideas are to keep the new policy close to the current one and to guarantee that it has a smaller cost than the current one. If the full state $\mathbf{s}_t$ is known and there is a single decision maker for the joint action $\mathbf{u}_t$, the ℓ-th TRPO update consists of first estimating the advantage function $A_{\pi_\ell}(\mathbf{s},\mathbf{u})$ of the current policy according to which the trajectories were generated, and then of solving[1]

$$\boldsymbol{\theta}_{\ell+1} = \arg\min_\theta \mathbb{E}\left[\frac{\pi(\mathbf{u}_t|\mathbf{s}_t;\boldsymbol{\theta})}{\pi(\mathbf{u}_t|\mathbf{s}_t;\boldsymbol{\theta}_\ell)}A_{\pi_\ell}(\mathbf{s}_t,\mathbf{u}_t)\right] \quad (8)$$

$$\text{s.t. } \mathbb{E}\left[D_{KL}(\pi(.|\mathbf{s}_t;\boldsymbol{\theta}_\ell)||\pi(.|\mathbf{s}_t;\boldsymbol{\theta}))\right] \le \delta. \quad (9)$$

Since TRPO can be cumbersome to implement in practice, PPO has been proposed in [21] as an approximation. The main idea is to enforce (9) by introducing a clipping function $g_\epsilon(.)$ with parameter $0 < \epsilon \ll 1$, and then to update the policy as

$$\boldsymbol{\theta}_{\ell+1} = \arg\min_\theta \mathbb{E}\left[\min\left(\frac{\pi(\mathbf{u}_t|\mathbf{s}_t;\boldsymbol{\theta})}{\pi(\mathbf{u}_t|\mathbf{s}_t;\boldsymbol{\theta}_\ell)}A_{\pi_\ell}, g_\epsilon(A_{\pi_\ell})\right)\right], \quad (10)$$

$$g_\epsilon(A) = \begin{cases}(1+\epsilon)A, & A \ge 0\\ (1-\epsilon)A, & A < 0.\end{cases} \quad (11)$$

The intuition behind PPO is that due to the clipping, there is no incentive for the updated policy to deviate too much from the old one. In practice, PPO works well and is considered as the state of the art in single agent RL problems.

The uplink scheduling setting we examine in this work involves multiple agents, each having a partial view of the system state at each slot t. In this case we use the local advantage function and thus, the TRPO for each BS is

$$\boldsymbol{\theta}^n_{\ell+1} = \arg\min_{\theta'}\mathbb{E}\left[\frac{\pi^n(u_n^t|\mathbf{o}_t^n;\boldsymbol{\theta}')}{\pi^n(u_n^t|\mathbf{o}_t^n;\boldsymbol{\theta}^n_\ell)}A^n_{\pi_\ell}(\mathbf{o}_n^t, u_n^t)\right] \quad (12)$$

$$\text{s.t. } \mathbb{E}\left[D_{KL}(\pi^n(.|\mathbf{o}_t^n;\boldsymbol{\theta}^n_\ell)||\pi^n(.|\mathbf{o}_t^n;\boldsymbol{\theta}'))\right] \le \delta. \quad (13)$$

This policy update is still distributed and for small values of δ it retains the property that the new joint policy is close to the current one, as suggested by the following proposition:

Proposition 2: Assume (13) for each BS n. Then for the joint policy we have

$$\mathbb{E}\left[D_{KL}(\pi(.|\mathbf{s}_t;\boldsymbol{\theta}_{\ell+1})||\pi(.|\mathbf{s}_t;\boldsymbol{\theta}_\ell))\right] \le N\delta.$$

[1]For the remainder, $D_{KL}(\mu||\mu') = \sum_x \mu(x)\log\left(\frac{\mu(x)}{\mu'(x)}\right)$ denotes the Kullback-Leibler divergence between probability distributions μ and μ'.

Proof: Denote the joint policy of agents $i, ..., N$ when the system state is $\mathbf{s}_t$ and the policy parameters of these respective agents are $\boldsymbol{\theta}_\ell^i, ..., \boldsymbol{\theta}_\ell^N$ as $\pi_\ell^{i:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_\ell^{i:N})$. In addition, $\pi_\ell^{n:n}(.|\mathbf{s}_t; \boldsymbol{\theta}_\ell^n) = \pi^n(.|\mathbf{o}_t^n; \boldsymbol{\theta}_\ell^n)$, and $\sum_{\mathbf{u}^{i:N}}$ is the sum over all possible joint actions of agents $i, ..., N$. For any state $\mathbf{s}_t$, from the KL divergence definition for a joint policy we have

$$
\begin{aligned}
&D_{KL}(\pi_{\ell+1}^{i:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_{\ell+1}^{i:N}) || \pi_\ell^{i:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_\ell^{i:N})) \\
&= \sum_{\mathbf{u}^{i:N}} \pi_{\ell+1}^{i:N}(\mathbf{u}^{i:N}|\mathbf{s}_t; \boldsymbol{\theta}_{\ell+1}^{i:N}) \sum_{j=i}^{N} \log\left(\frac{\pi_{\ell+1}^j(u^j|\mathbf{o}_t^j; \boldsymbol{\theta}_{\ell+1}^j)}{\pi_\ell^j(u^j|\mathbf{o}_t^j; \boldsymbol{\theta}_\ell^j)}\right) \\
&\overset{(a)}{=} \sum_{u^i} \pi_{\ell+1}^i(u^i|\mathbf{o}_t^i; \boldsymbol{\theta}_{\ell+1}^i) \log\left(\frac{\pi_{\ell+1}^i(u^i|\mathbf{o}_t^i; \boldsymbol{\theta}_{\ell+1}^i)}{\pi_\ell^i(u^i|\mathbf{o}_t^i; \boldsymbol{\theta}_\ell^i)}\right) + \\
&\sum_{\mathbf{u}^{i+1:N}} \pi_{\ell+1}^{i+1:N}(\mathbf{u}^{i+1:N}|\mathbf{s}_t; \boldsymbol{\theta}_{\ell+1}^{i+1:N}) \sum_{j=i+1}^{N} \log\left(\frac{\pi_{\ell+1}^j(u^j|\mathbf{o}_t^j; \boldsymbol{\theta}_{\ell+1}^j)}{\pi_\ell^j(u^j|\mathbf{o}_t^j; \boldsymbol{\theta}_\ell^j)}\right) \\
&= D_{KL}(\pi_{\ell+1}^i(.|\mathbf{o}_t^i; \boldsymbol{\theta}_{\ell+1}^i) || \pi_\ell^i(.|\mathbf{o}_t^i; \boldsymbol{\theta}_\ell^i)) \\
&\quad + D_{KL}(\pi_{\ell+1}^{i+1:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_{\ell+1}^{i+1:N}) || \pi_\ell^{i+1:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_\ell^{i+1:N})),
\end{aligned}
$$

where in step (a) the properties $\sum_{u^i} \pi_{\ell+1}^i(u^i|\mathbf{o}_t^i; \boldsymbol{\theta}_{\ell+1}^i) = \sum_{\mathbf{u}^{i+1:N}} \pi_{\ell+1}^{i+1:N}(\mathbf{u}^{i+1:N}|\mathbf{s}_t; \boldsymbol{\theta}_{\ell+1}^{i+1:N}) = 1$ are used, which hold since these sums are probability distributions. By taking the expectation over the state distribution and using (13), we get

$$
\begin{aligned}
&\mathbb{E}\left[D_{KL}(\pi_{\ell+1}^{i:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_{\ell+1}^{i:N}) || \pi_\ell^{i:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_\ell^{i:N}))\right] \\
&\leq \delta + \mathbb{E}\left[D_{KL}(\pi_{\ell+1}^{i+1:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_{\ell+1}^{i+1:N}) || \pi_\ell^{i+1:N}(.|\mathbf{s}_t; \boldsymbol{\theta}_\ell^{i+1:N}))\right],
\end{aligned}
$$

and then applying it recursively completes the proof. ∎

Moreover, it can be shown that (12) corresponds to a BS that updates its policy in order to optimize the global cost, assuming that the policies of all other BSs do not change. Since the new individual policies, as well as the new joint policy as shown in Proposition 2, are close to the old ones, we expect that the joint policy will on average get updated towards a direction which improves the long term cost.

In order to replace (12)-(13) by PPO, we approximate the expectations by a Monte-Carlo estimation from M episodes of T slots, as in VPG-GC. By using (7) for the advantage estimation, the parameters of the policy of BS n are updated such that the following surrogate cost is minimized:

$$
L^n(\boldsymbol{\theta}) = \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=0}^{T-1} \min_{\theta^n} \left(\frac{\pi^n(u_t^n|\mathbf{o}_t^n; \boldsymbol{\theta}^n)}{\pi^n(u_t^n|\mathbf{o}_t^n; \boldsymbol{\theta}_\ell^n)} \hat{A}_{m,t}^n, g_\epsilon(\hat{A}_{m,t}^n) \right). \tag{14}
$$

Finally, since clipping alone does not guarantee that the updated policy is close to the old one, we also require that the updated policy satisfies a Monte-Carlo based estimate of (13):

$$
\frac{1}{MT} \sum_{m=1}^{M} \sum_{t=0}^{T-1} \log\left(\frac{\pi^n(u_t^n|\mathbf{o}_t^n; \boldsymbol{\theta}_\ell^n)}{\pi^n(u_t^n|\mathbf{o}_t^n; \boldsymbol{\theta}^n)}\right) \leq \delta. \tag{15}
$$

We enforce the above by updating $\boldsymbol{\theta}_\ell^n$ until (15) is violated. The whole process is summarized in Alg. 2.

The algorithms presented in this section can be implemented in a distributed manner, for any observation structure, as long as each BS has access to the total cost $\sum_k w^k x_t^k$ at the beginning of each slot. This information is easy to disseminate through the control interface connecting the BSs, as each BS has to broadcast only a real number (the part of the cost that corresponds to its devices). It is expected therefore that the algorithms will converge towards scheduling policies whose joint policy is a local minimum of the control problem (4). The policy and its performance depend on the information sharing pattern, i.e. the BSs $\mathcal{N}(n)$ that BS n can exchange information. We investigate this dependence in the numerical results. Finally, a main aspect of the presented algorithms is that the policies stay constant for a period of time. In contrast to independent deep Q-learning [22], where the policies are updated at every time step, this allows the agents to view the environment as stationary during policy evaluation and update their policies towards a good direction.

Algorithm 2 Global Cost PPO (PPO-GC)

Input: Maximum allowed KL divergence δ, clipping ratio ϵ, GAE parameter β, initialize $\boldsymbol{\theta}_0^n, \boldsymbol{\phi}^n$

for $\ell = 0, 1, 2, \dots$ **do**:
  for $m = 1, 2, \dots, M$ **do**:
    Run policy with $\boldsymbol{\theta}_\ell^n$ and store the trajectory of local observations $\left\{(\mathbf{o}_{m,t}^n, u_{m,t}^n, C_{m,t})\right\}_{t=0,..,T-1}$
    Calculate $\{\hat{A}_{m,t}^n\}_{t=0,...,T-1}$ according to (7)
  $\boldsymbol{\theta}_{\ell+1}^n \leftarrow \boldsymbol{\theta}_\ell^n$
  while not converged **do**:
    Update $\boldsymbol{\theta}_{\ell+1}^n$ with one gradient descent step (14)
    if (15) is not satisfied **then**:
      Set $\boldsymbol{\theta}_{\ell+1}^n$ to its value before last gradient step
      break
  $\boldsymbol{\phi}^n \leftarrow \arg\min_{\phi'} \frac{1}{MT} \sum_m \sum_t \left(V_{\phi'}^n(\mathbf{o}_t^n) - C_t\right)^2$

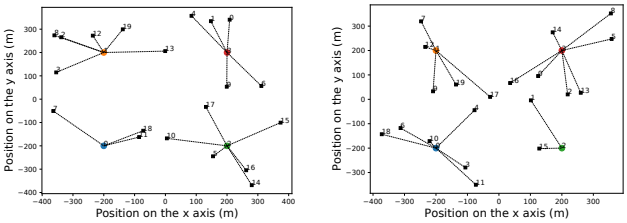
IV. Simulation Results

In this section, we illustrate the performance of the previously presented MARL algorithms for uplink scheduling. We investigate (i) how MARL algorithms compare against standard schedulers, (ii) the impact of the method used for policy gradient (VPG vs. PPO) and (iii) how does state information sharing affect convergence behavior and performance. We use the following baselines:

- *Proportional Fair Scheduler (PF)*: BS n schedules
$$
u_t^{n,\text{PF}} = \arg\max_{k \in \mathcal{K}(n)} \left\{ \frac{1}{\bar{R}_t^k} \log_2\left(1 + g_t^{k,n} P\right) \right\},
$$
where $\bar{R}_t^k$ is the average rate of device k (see e.g. [1]).
- *MaxWeight Scheduler (MW)*: BS n schedules
$$
u_t^{n,\text{MW}} = \arg\max_{k \in \mathcal{K}(n)} \left\{ x_t^k \log_2\left(1 + g_t^{k,n} P\right) \right\}.
$$
This scheduler takes into account both channel conditions and buffer length of the device, and aims to stabilize the latter; see [2] for more details.
- *Independent PPO (iPPO)*: A PPO based algorithm where each BS uses only the local part of the cost, i.e. the queue lengths and channel gains of its associated devices.

We assume $N = 4$ BSs in a square control topology. Traffic follows a Poisson distribution with mean $\lambda_k = \lambda, \forall k$ and the weights of the cost function (4) are equal to $w^k = 1, \forall k$. The local state of each BS consists of queue lengths and channel gains of its devices. We denote as *1hop* the setting where only neighboring BSs share their local state information and as *2hop* the setting with full state information. In all other cases, a BS observes only its local state. Both policy and local value function estimation, referred to as actor and critic respectively, are modeled by neural networks with fully connected layers. For the wireless channel we assume a dual-slope path loss model [12], lognormal shadowing and a Gauss-Markov model for fast fading. We generate 10 topologies with $K = 20$ randomly placed devices, each one associated to the closest BS. We observe the accumulated total reward per epoch over 300 training epochs as well as the evolution of the sum of queue lengths in an episode after training. For every topology, each experiment is run 5 times with different realization of neural networks and channel and traffic processes. To improve numerical stability, we normalize the queue states with the system's bandwidth, i.e. the related input to the neural network of device k is x_t^k/W. An overview of the parameters is shown in Table I. For evaluation purposes, we focus on:

(a) *Learning behavior*: total cost accrued for each training epoch for arrival rate $\lambda = 150$ bits/slot for each device.
(b) *Performance*: evolution of the average normalized total queue length $\frac{1}{KW}\sum_{k=1}^{K} x_t^k$ over time for the above arrival rate after training.
(c) *Robustness*: normalized mean total queue length of devices $\frac{1}{TW}\sum_t \sum_k x_t^k$ after training for different λ. Higher arrival rate means that the system is more loaded, therefore coordination among BSs is more critical.



(a) Topology 1 (b) Topology 2

Fig. 2: Representative network topologies. Dotted lines represent the device association to BSs.

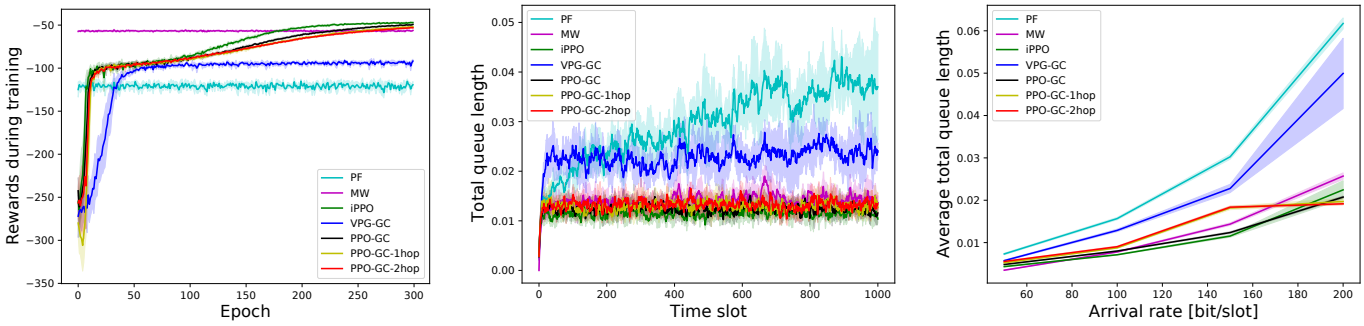
TABLE I: Simulations parameters.

Environment parameters	
Number of devices, Number of BS	20, 4
Distance between BS, Breakpoint distance	400m, 100m
Bandwidth W, Timeslot duration T_s	1MHz, 1ms
Transmission Power (before normalization)	-10dBm
Noise power spectral density	-174dBm/Hz
Path loss exponents	2 (near), 4 (far)
Path loss constant, Shadowing standard deviation	39dB, 7dB
Fast fading correlation	0.8
Algorithm parameters	
Hidden Layers of Actor/Critic Neural Network	$\{64, 64\}$
Activation function	tanh
Learning rate a for Algorithm 1 and critic (6)	3×10^{-4}, 10^{-3}
GAE parameter β, Discount factor γ	0.97, 0.99
Clipping ratio ϵ, Maximum KL divergence δ	0.2, 0.01
Training Iterations for critic (6) and PPO (14)	80
Number of episodes per training epoch M	4
Episode length T	1000 slots
Number of Topologies, Number of random seeds	10, 5

Two representative topologies are depicted in Fig. 2. Topology 1 in Fig. 2(a) is an "easy" setting with few cell edge devices; this implies that intercell interference is low, therefore the BS decisions do not have a big impact to each other. In contrast, Topology 2 in Fig. 2(b) has many edge devices, which makes the impact of a BS action to other BSs more significant due to higher interference. The results regarding the previous performance metrics are shown in Fig. 3-4 for Topology 1 and 2, respectively. We can see that PPO-GC performs much better than VPG-GC when every BS has local state observation. For instance, VPG-GC may get stuck in a bad local minimum as in Fig. 3(a) or even fail to consistently improve performance, as in Fig. 4(a). This is what happens in single agent RL, where PPO is experimentally shown to outperform VPG, and suggests that keeping the updated policy close to the old one is beneficial for multiple agents as well. Moreover, we observe that all PPO-GC algorithms outperform the baselines. This is aligned to our intuition from Theorem 1 that a policy gradient-based RL algorithm with global cost performs updates in a good direction. In addition, PF is the worst-performing algorithm in all topologies, but this is expected since it does not take into account the queue lengths of the devices.

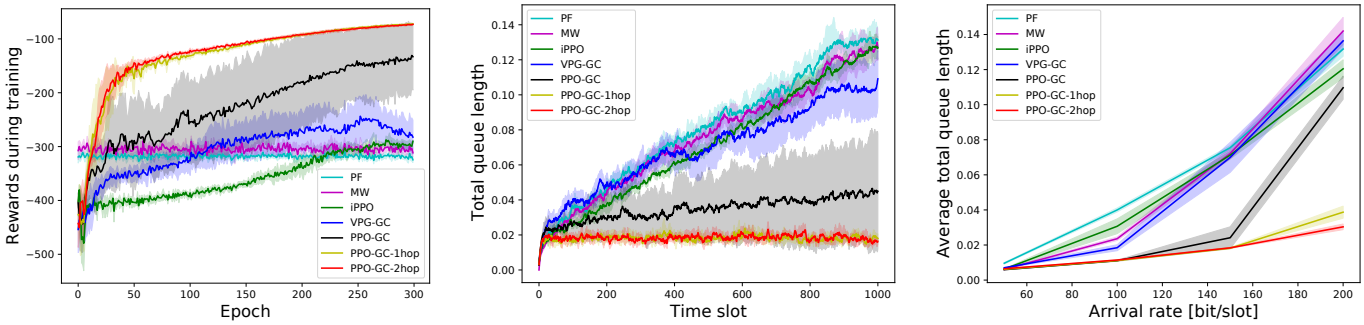
The relative performance among PPO-GC algorithms depends on the problem instance on a consistent way. In topologies where MW has a good performance, there is no much difference and even iPPO performs well. On the other hand, when the system is unstable under MW, iPPO also performs bad, while PPO-GC algorithms are better and information sharing is beneficial. Indeed, we can see in Fig. 3(a)-3(b) that all learning algorithms have similar performance for Topology 1 and MW keeps the total cost low. The situation is very different for Topology 2 in Fig. 4(a)-4(b), where the system is unstable under both iPPO and MW. PPO-GC learns a better policy, while PPO-GC-1hop and PPO-GC-2hop are far superior. The reason is that there are more edge devices among the BSs in Topology 2 and thus, common information in the form of the global cost becomes crucial to learn the impact of local actions to other BSs. The exchange of state information enables the BSs to take even more coordinated decisions.

Regarding robustness, we see in Fig. 3(c)-4(c) that for both topologies there are gains from information sharing when the network is highly loaded. This is more evident for Topology 2. where the high load regime starts from small arrival rates. Finally, the algorithms with information sharing perform worse for low arrival rates in Topology 1, most likely due to the fact that they have more parameters and need more training steps.



(a) Learning behavior (b) Performance (c) Robustness

Fig. 3: Results for Topology 1 (averaged over 5 runs; the shaded area corresponds to one standard deviation).



(a) Learning behavior (b) Performance (c) Robustness

Fig. 4: Results for Topology 2 (averaged over 5 runs; the shaded area corresponds to one standard deviation).

V. Conclusions

In this paper we provided a decentralized learning algorithm for BSs to find a distributed policy for the uplink scheduling of devices. The proposed procedure requires very low signalling and results in policies that outperform standard schedulers. In addition, numerical results show that exchanging additional state information helps the performance in heavy traffic scenarios and/or scenarios with many devices at the cell edge.

References

[1] M. Andrews *et al.*, "Providing quality of service over a shared wireless link," *IEEE Commun. Mag.*, vol. 39, no. 2, pp. 150–154, 2001.

[2] ——, "Scheduling in a Queueing System with Asynchronously Varying Service Rates," *Probability in Engineering and Information Sciences*, vol. 18, no. 2, pp. 191—217, Apr. 2004.

[3] B. Rengarajan and G. de Veciana, "Architecture and Abstractions for Environment and Traffic-Aware System-Level Coordination of Wireless Networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 721–734, 2011.

[4] J. Niu *et al.*, "Multi-cell cooperative scheduling for uplink sc-fdma systems," in *IEEE PIMRC*, 2013.

[5] G. Venkatraman, A. Tölli, M. Juntti, and L.-N. Tran, "Traffic Aware Resource Allocation Schemes for Multi-Cell MIMO-OFDM Systems," *IEEE Trans. Signal Process.*, vol. 64, no. 11, pp. 2730–2745, 2016.

[6] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, 2021.

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

[8] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multi-user cellular networks: Deep reinforcement learning approaches," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6255–6267, 2020.

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

[10] L. Zhang and Y.-C. Liang, "Deep reinforcement learning for multi-agent power control in heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 4, pp. 2551–2564, 2021.

[11] H. Lee and J. Jeong, "Multi-agent deep reinforcement learning (madrl) meets multi-user mimo systems," *IEEE GLOBECOM*, 2021.

[12] N. Naderializadeh, 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.

[13] J. Pajarinen, A. Hottinen, and J. Peltonen, "Optimizing spatial and temporal reuse in wireless networks by decentralized partially observable markov decision processes," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 866–879, 2014.

[14] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, 2019.

[15] S. Moon *et al.*, "Neuro-DCF: Design of Wireless MAC via Multi-Agent Reinforcement Learning Approach," in *ACM MobiHoc*, 2021.

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

[17] L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling, "Learning to Cooperate via Policy Search," in *16th Conf. Uncertain. Artif. Intell. UAI*, 2000.

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[19] J. Schulman *et al.*, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," in *ICLR*, 2016.

[20] ——, "Trust Region Policy Optimization," in *ICML*, 2015.

[21] ——, "Proximal Policy Optimization Algorithms," arXiv:2011.09533 [cs.LG], 2017.

[22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, p. 529–533, 2015.