Model-free Decentralized Training for Deep Learning Based Resource Allocation in Communication Networks

Pourya Behmandpoor, Panagiotis Patrinos, and Marc Moonen
KU Leuven, Department of Electrical Engineering (ESAT),
STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics.

***Abstract*—Decentralized deep learning (DL) based resource allocation (RA) in communication networks guarantees scalability and higher communication bandwidth efficiency compared to centralized RA. Although the RA is decentralized in such approaches, the policies are mostly trained in a centralized manner. In this paper, we investigate a decentralized model-free training approach based on zeroth-order optimization methods. Each user trains its individual policy—possibly with a unique structure—to guarantee the maximum global utility, e.g., sum rate (SR) of users. More importantly, during the training, the users need to share only scalar quantities with their neighbors, avoiding a large communication overhead. The training is also robust against a certain level of asynchrony between the users. The proposed approach relaxes the need for a computationally complex central server and offers the possibility for (re)training in dynamic environments in a model-free manner using the computational power at the edge. Numerical experiments show a competitive performance compared to centralized and federated training approaches.**

***Index Terms*—Decentralized training, decentralized resource allocation, deep learning, model-free, asynchronous, communication networks**

I. Introduction

Deep learning (DL) based resource allocation (RA) in communication networks has changed the paradigm of direct design of RA techniques to devising efficient policies, e.g., deep neural networks (DNNs), and optimization methods to train such policies in order to achieve simplicity in implementation and efficiency in performance compared to the conventional approaches [1].

DL based RA consists of two main stages: the training stage and the inference stage. In the training stage, the policy learns RA via maximizing an appropriate reward function in either a supervised or unsupervised manner, by different policy training approaches, e.g., reinforcement learning (RL) [2]–[14].

Once the policy is trained, in the inference stage, RA is performed in a centralized or decentralized manner depending on whether there is a server assigned to perform such a task or each user performs RA by exploiting its local policy [2], [6], [15]. In this stage, RA approaches are of interest that are fast, incur less communication overhead, and can generalize, e.g., on different communication network sizes.

While there has been extensive research on DL based RA methods with a decentralized inference stage, the training is mostly done in a centralized manner [6], [15]. Decentralized training, though, can have several advantages: online training of policies after pre-training helps the policies to capture real-world parameters of the communication network in real-time, e.g., channel distribution, number of users, and user demands. Such online training may also be repeated once in a while during the operation to tackle drifts from the initial status of the communication network, e.g., channel distribution drift when the channel is nonstationary [16], [17]. Since the measurements, e.g., for channel estimation, are done locally by each user, sending the local measurements to a central server to perform such (re)training tasks incurs a communication overhead and additional delays. Moreover, a server performing such a training task needs to have enough computation and communication capacity, which may be absent in real-world scenarios.

Learning over graphs and federated learning (FL) [18], [19] are two optimization frameworks that recently have received a lot of attention in distributed learning. In distributed policy training for RA [20], [21], a server is mostly required to lead the training and to guarantee synchrony between the users. Also, these approaches mostly rely on sharing parameters such as gradient vectors between the users which again incurs a communication overhead. Therefor, in this paper, a model-free asynchronous decentralized training approach is presented for the existing DL based RA methods. Each user has an individual policy and needs to send only scalar quantities to other users once in a while, avoiding communication overhead.

II. System model

We consider N users (communication links) each equipped with a transmitter and a receiver. The transmit power of user i is denoted by $p_i \in [0, p_i^{max}]$, the ith element of the vector $\boldsymbol{p} \in P_C$, where $P_C \coloneqq \{\boldsymbol{p} = (p_1, \ldots, p_N) \in \mathbb{R}^N \mid p_i \in [0, p_i^{max}],\ \forall i \in [N] \coloneqq \{1, \ldots, N\}\}$. The direct channel between the transmitter and receiver of user i is denoted by h_{ii}, while the interference channel between the transmitter of user j and the receiver of user i is denoted by h_{ij}. All the channel coefficients define the full channel matrix $\boldsymbol{H} \in \mathbb{C}^{N \times N}$ with

This research work was carried out at the ESAT Laboratory of KU Leuven, in the frame of Research Project FWO nr. G0C0623N 'User-centric distributed signal processing algorithms for next generation cell-free massive MIMO based wireless communication networks' and Fonds de la Recherche Scientifique - FNRS and Fonds voor Wetenschappelijk Onderzoek - Vlaanderen EOS Project no 30452698 '(MUSE-WINET) MUlti-SErvice WIreless NETworks'. The scientific responsibility is assumed by its authors.

h_{ij} its element in the ith row and jth column. The channel has a distribution $\mathcal{D}_H$ with support Ω_H. The additive white Gaussian noise in each receiver is assumed to be independent and identically distributed (IID) with power σ_n^2, which is assumed to be constant and the same in all receivers.

The achievable data rate of user i, $R_i : \mathbb{C}^{N\times N} \times P_C \to \mathbb{R}_+$, can then be written as,

$$R_i(\boldsymbol{H}, \boldsymbol{p}) \coloneqq \log_2 \left(1 + \frac{|h_{ii}|^2 p_i}{\sigma_n^2 + \sum_{j\neq i} |h_{ij}|^2 p_j}\right). \tag{1}$$

We consider an individual policy e.g., a DNN, for user $i \in [N]$ as $\phi_i : \mathbb{R}^{m_i} \times \boldsymbol{\Theta}_i \to \mathbb{R}_+$. The policy $\phi_i(\boldsymbol{s}_i, \boldsymbol{\theta}_i)$ outputs the resource allocated to user i, e.g., transmit power, subchannel, etc. The policy input is the vector $\boldsymbol{s}_i \in \mathbb{R}^{m_i}$ including local measurements of user i, and the policy parameter vector is $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}_i \subset \mathbb{R}^{n_i}$, including e.g., DNN weights and biases. Note that each user i may have its unique local measurements $\boldsymbol{s}_i$, which depend on the channel $\boldsymbol{H}$. The vector $\boldsymbol{s}_i$ may contain the direct channel, interference power, noise power, etc. (cf. Section IV).

The RA considered here corresponds to the following optimization of the policy parameters $\boldsymbol{\theta}_i$ to maximize the global utility, equal to the sum of individual utilities $U = \sum_{i=1}^N U_i$, while satisfying individual box constraints:

$$\boldsymbol{\theta}^\star \in \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_{i=1}^N \mathbb{E}_{\boldsymbol{H}\sim\mathcal{D}_H}\{U_i(\boldsymbol{H}, \boldsymbol{\phi}(\boldsymbol{H}, \boldsymbol{\theta}))\} \tag{2}$$
$$\text{subject to } \phi_i(\boldsymbol{s}_i, \boldsymbol{\theta}_i) \in C_i,\ \forall i \in [N], \forall \boldsymbol{H} \in \Omega_H.$$

Here $\boldsymbol{\theta} \coloneqq (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N) \in \mathbb{R}^{n:=\sum n_i}$, $\boldsymbol{\phi}(\boldsymbol{H}, \boldsymbol{\theta}) \coloneqq (\phi_1(\boldsymbol{s}_1, \boldsymbol{\theta}_1), \ldots, \phi_N(\boldsymbol{s}_N, \boldsymbol{\theta}_N))$, and C_i is the resource box constraint. Also, U_i is the utility function of user i, e.g., $U_i = R_i(\boldsymbol{H}, \boldsymbol{\phi}(\boldsymbol{H}, \boldsymbol{\theta}))$ or $U_i = \log R_i(\boldsymbol{H}, \boldsymbol{\phi}(\boldsymbol{H}, \boldsymbol{\theta}))$.

Once the policies are trained, each user monitors its local policy output to adjust its resource accordingly, e.g., $p_i = \phi_i(\boldsymbol{s}_i, \boldsymbol{\theta}_i^\star)$ with $C_i = [0, p_{max}]$.

III. Proposed Decentralized Training

The training in DL based RA is mostly done in a centralized manner performed by a central server that synchronously gathers all the necessary information, i.e., the vectors $\boldsymbol{s}_i$ and $\boldsymbol{H}$, and performs updates on $\boldsymbol{\theta}$ until convergence. Define

$$f^\phi(\boldsymbol{H}, \boldsymbol{\theta}) \coloneqq \textstyle\sum_{i=1}^N U_i(\boldsymbol{H}, \boldsymbol{\phi}(\boldsymbol{H}, \boldsymbol{\theta}))$$
$$\bar{f}^\phi(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{\boldsymbol{H}\sim\mathcal{D}_H}\{f^\phi(\boldsymbol{H}, \boldsymbol{\theta})\}.$$

The stochastic gradient ascent (SGA) method can be used to iteratively update the policy parameter $\boldsymbol{\theta}$ by $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \gamma^k \frac{1}{B} \sum_{\ell=1}^B \nabla f^\phi(\boldsymbol{H}^\ell, \boldsymbol{\theta}^k)$. Here, k is the iteration number, B is the size of a minibatch of channel instances drawn from the distribution $\mathcal{D}_H$ over B time instants, and γ^k is a diminishing stepsize to guarantee convergence.

One of the required quantities in SGA is the gradient vector which should be known to the optimizer. Note that this vector has high dimensionality, i.e., $n = \sum n_i$. Hence, sharing this vector quantity between the users and the server, or the users with their peers, imposes a large communication overhead. Moreover, deriving such a gradient vector requires explicit knowledge about the utility function and all the parameters involved, and cannot consider unknown nonidealities in the communication network such as nonlinearities in the transmitters and the receivers [4].

Zeroth-order optimization approaches [22] optimize the policy parameters by only evaluating function values. In RA, this function evaluation may correspond to measuring, e.g., user data rates in a model-free manner (cf. e.g., [4]), involving all unknown nonidealities. Before employing the zeroth-order optimization approach, a required assumption is as follows:

Assumption I. *The policies $\phi_i(\boldsymbol{s}_i, \cdot)$, $i \in [N]$, are continuous for all $\boldsymbol{s}_i \in \mathbb{R}^{m_i}$, e.g., they are continuous DNNs.*

A so-called *smoothed approximation function* $\bar{f}^{\phi,\mu} : \mathbb{R}^n \to \mathbb{R}_+$ is then defined as

$$\bar{f}^{\phi,\mu}(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{\boldsymbol{u}}\{\bar{f}^\phi(\boldsymbol{\theta} + \mu\boldsymbol{u})\} \quad \text{where } \boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \mathrm{I}),\ \mu > 0. \tag{3}$$

This function has interesting properties, two of which are highlighted as follows [22]:

Proposition III.1 (smoothed approximation function $\bar{f}^{\phi,\mu}$)**.**
(i) Let $\bar{f}^\phi$ be continuous and $\mu > 0$, then $\bar{f}^{\phi,\mu}$ is Lipschitz differentiable;
(ii) $\nabla \bar{f}^{\phi,\mu}(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{\boldsymbol{u}}\{\frac{\bar{f}^\phi(\boldsymbol{\theta}+\mu\boldsymbol{u}) - \bar{f}^\phi(\boldsymbol{\theta})}{\mu}\boldsymbol{u}\}$.

Note that the name *smoothed approximation function* is due to Proposition III.1*(i)*, as it indicates that even if the original function $\bar{f}^\phi$ is not differentiable, the function $\bar{f}^{\phi,\mu}$ is Lipschitz differentiable, i.e., a smoothed approximation of the original function. Moreover, Proposition III.1*(ii)* also defines the gradient of the smoothed approximation function employing expected function values over the random vector $\boldsymbol{u}$. This gradient is used in the zeroth-order update of the parameter $\boldsymbol{\theta}$, i.e.,

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \gamma^k \hat{\nabla} \bar{f}^{\phi,\mu}(\boldsymbol{\theta}^k), \tag{4}$$

where $\hat{\nabla}\bar{f}^{\phi,\mu}$ is an empirical estimation of $\nabla\bar{f}^{\phi,\mu}$ by one or a minibatch of the random vectors $\boldsymbol{u}$. The zeroth-order update rule (4) is convergent under Assumption I and also some other regularity conditions, up to a certain precision dependent on the parameter μ [22, Sec. 7].

A. Decentralized updates

In the zeroth-order update rule (4), the estimated gradient $\hat{\nabla}\bar{f}^{\phi,\mu}$ is evaluated at $\boldsymbol{\theta}^k$ by measuring the function values of $\bar{f}^\phi$ at $\boldsymbol{\theta}^k$ and at its randomly perturbed version $\boldsymbol{\theta}^k + \mu\boldsymbol{u}^k$. Note that the function $\bar{f}^\phi$ can be estimated using minibatches of channel instances over a period of time. Hence, the update rule (4) is recast as

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \gamma^k \alpha^k \boldsymbol{u}^k, \tag{5}$$
$$\alpha^k \coloneqq \frac{\mathbb{E}_{\boldsymbol{H}}^{B_2}\{f^\phi(\boldsymbol{H}, \boldsymbol{\theta}^k + \mu\boldsymbol{u}^k)\} - \mathbb{E}_{\boldsymbol{H}}^{B_1}\{f^\phi(\boldsymbol{H}, \boldsymbol{\theta}^k)\}}{\mu}$$

where two minibatches of channel instances with size B_1 and B_2 are used and $\mathbb{E}_H^{B_i}\{\cdot\}$ denotes the average over the minibatch with size B_i. Two different minibatches allow the users to practically measure the utility function for $\boldsymbol{\theta}^k$ and $\boldsymbol{\theta}^k + \mu\boldsymbol{u}^k$ in two distinct time periods. To do so, each user estimates its own average utility in two successive time periods to form

$$\alpha^k = \tfrac{1}{\mu} \textstyle\sum_{i=1}^N d_i^k \tag{6}$$

Algorithm 1 Proposed decentralized training

Initialize: $\gamma_0 > 0$ and $\tilde{\gamma} > 0$ for $\gamma^k = \gamma_0/(k+1)^{\tilde{\gamma}}$, $B_1 > 0$, $B_2 > 0$, $\mu > 0$, $\boldsymbol{\theta}_i^0 \ \forall i \in [N]$

each user i **asynchronously performs:**

for $k = 0, 1, \ldots,$

1: continuously receive d_j $j \neq i$ from the other users
2: measure $\mathbb{E}_{\boldsymbol{H}}^{B_1}\{U_i\}$
3: draw a random vector $\boldsymbol{u}_i^k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
4: update the policy $\phi_i(\cdot, \boldsymbol{\theta}_i^k + \mu \boldsymbol{u}_i^k)$
5: measure $\mathbb{E}_{\boldsymbol{H}}^{B_2}\{U_i\}$
6: calculate d_i^k using (6) and send it to the other users
7: perform update on $\boldsymbol{\theta}_i^k$ using (8)
8: update the policy $\phi_i(\cdot, \boldsymbol{\theta}_i^{k+1})$

$$\text{where} \quad d_i^k := \mathbb{E}_{\boldsymbol{H}}^{B_2}\{U_i(\boldsymbol{H}, \boldsymbol{\phi}(\boldsymbol{H}, \boldsymbol{\theta}^k + \mu \boldsymbol{u}^k))\} - \mathbb{E}_{\boldsymbol{H}}^{B_1}\{U_i(\boldsymbol{H}, \boldsymbol{\phi}(\boldsymbol{H}, \boldsymbol{\theta}^k))\}.$$

The scalar quantity α^k can be made available at all the users by message passing, i.e., each user sending its own estimate d_i^k as a scalar to all other users. Once α^k is known at all the users, the update rule (5) can be recast as

$$\begin{aligned} \boldsymbol{\theta}_i^{k+1} &= \boldsymbol{\theta}_i^k + \gamma^k \alpha^k \boldsymbol{u}_i^k \quad \forall i \in [N], \\ &= \boldsymbol{\theta}_i^0 + \tfrac{1}{\mu} \textstyle\sum_{t=0}^{k} \sum_{j=1}^{N} \gamma^t d_j^t \boldsymbol{u}_i^t, \end{aligned} \tag{7}$$

where $\boldsymbol{\theta}_i$ is defined in (2). By the decentralized update rule in (7), the users only share one scalar quantity α^k every $B := B_1 + B_2$ time instants.

B. Asynchronous updates

We also consider the case where the users have asynchronous local updates and transmissions during the decentralized training. It is assumed that the users have the same update frequency, i.e., it takes an equal amount of time for each user to have B samples. Without loss of generality, we sort the users from earliest to latest, hence, user 1 has its updates earlier than any other user $j > 1$. The asynchronous updating is then shown in Fig. 1. In this setting, the local update for each user i can be specified as

$$\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \tfrac{1}{\mu}\Big[\gamma^k \textstyle\sum_{j \leq i} d_j^k \boldsymbol{u}_i^k + \gamma^{k-1} \sum_{j>i} d_j^{k-1} \boldsymbol{u}_i^{k-1}\Big]. \tag{8}$$

Note that user i needs to form α^k from all d_i^k $\forall i \in [N]$ as in (7). However, at the time of the update, the slower users $j > i$ have not yet sent their updates on d_j^k. Instead, user i can use d_j^{k-1}, $j > i$ from the slower users, received at the previous iteration $k-1$. Moreover, the updates d_j^{k-1} correspond to $\boldsymbol{u}_i^{k-1}$ as this is the case in (7). By setting $d_j^{-1} = 0$ for $j > i$ and $\boldsymbol{u}_i^{-1} = 0$, the update rule (8) has the same summands as the update rule in (7), except for the summands corresponding to $d_j^k, j > i$ that are not yet received by user i at iteration k. Note that to guarantee an accurate enough gradient estimation, all the users should have their parameters $\boldsymbol{\theta}_i^k$ and the perturbed versions $\boldsymbol{\theta}_i^k + \mu \boldsymbol{u}_i^k$ at the same time—or with a small misalignment—in the function f^ϕ in (5). Hence, as an assumption, we require to have a bounded maximum misalignment between the users, i.e., if $B_1 = B_2$, the maximum misalignment requires to be less than $B/2$ time instants.

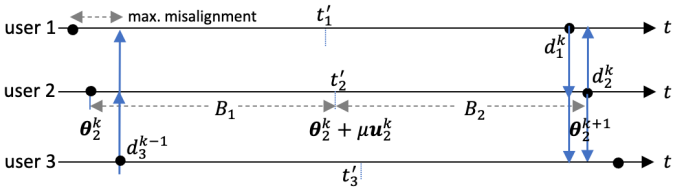

Figure 1: *A 3-user scenario with the global time t. The bullets are the update moments of the users. Each user needs $B = B_1 + B_2$ samples over B time instants in each iteration. The vertical arrows represent message passing between the users. Each user sends its d_i^k to the other users asynchronously. User 2 uses d_3^{k-1}, d_1^k, and d_2^k in its update (8).*

Each user i also needs to know if the received d_j belongs to the current iteration k or the previous iteration $k-1$. This can be simply deduced in each user based on whether d_j is received during or after the first B_1 samples (cf. t_i' in Fig. 1).

The proposed training is summarized in Algorithm 1. Note that although U_i depends on all the policy parameters $\boldsymbol{\theta}$ and the random vectors $\boldsymbol{u}$, each user only needs to measure its own average utility without knowing these parameters. For instance, if the user utility is a function of its data rate, this measurement may be performed by estimating the direct channel and the received interference plus noise power in (1) or by any other data rate estimation method [4]. Moreover, the local policies may be different after convergence, i.e., $\boldsymbol{\theta}_i \neq \boldsymbol{\theta}_j, i \neq j$, providing more flexibility for the decentralized RA to reach the maximum global utility.

C. Connected graph

The proposed approach can be extended to the case where each user i only communicates with its neighbors $\mathcal{N}_i \subseteq [N]$ instead of with all the users. The network is represented by a graph, with vertices representing the users and edges with values 0 or 1, specifying if the corresponding users are connected or not. The underlying graph is *connected*, i.e., any two vertices are linked in both directions either directly or by intermediate vertices. In this case, each user i sends its measurements d_i with a timestamp $\mathcal{T}_i$ and a user ID to its neighbors. The users save the received quantities in their buffer $\mathcal{B}_i$ and send these quantities to their neighbors in the next communication instant. The buffer is large enough so each user can save the scalars d_j from all the users in the network. Also, the old entries are discarded when the buffer is full. The update rule of (8) is then extended to

$$\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \tfrac{\gamma^k}{\mu} \textstyle\sum_{j \in \mathcal{B}_i} d_j^{\mathcal{T}_j^i(k)} \boldsymbol{u}_i^{\mathcal{T}_j^i(k)}, \tag{9}$$

where $\mathcal{T}_j^i(k)$ is the timestamp of user j available at user i at iteration k. Note that $\mathcal{T}_j^i(k) \leq k$ and $\mathcal{T}_i^i(k) = k$ for all $i \in [N]$. Moreover, each user i has a second buffer for saving its own generated perturbation vectors $\{\boldsymbol{u}_i^k, \boldsymbol{u}_i^{k-1}, \ldots, \boldsymbol{u}_i^{k-D_{max}}\}$, with D_{max} as the maximum delay in the graph.

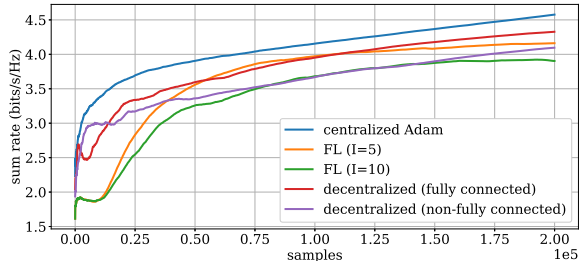

Figure 2: *Performance comparisons with the centralized and FL-based approaches, in a 5-user scenario.*

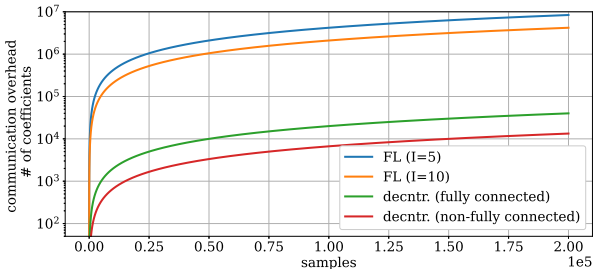

Figure 3: *Communication overhead comparisons with the centralized and FL-based approaches, in a 5-user scenario.*

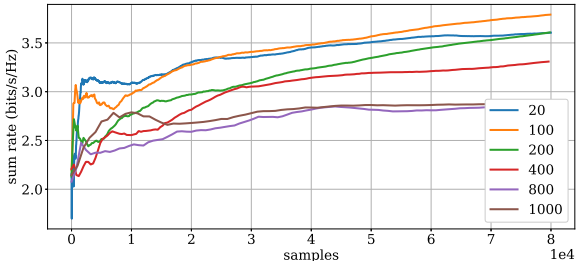

Figure 4: *Performance of the proposed approach with different batch sizes B, in a 5-user scenario.*

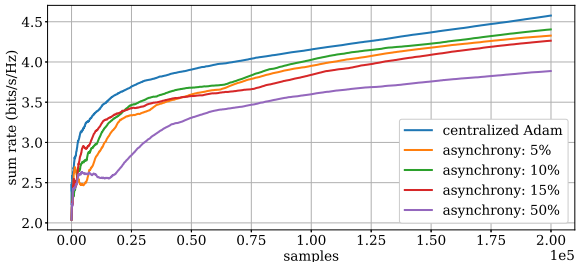

Figure 5: *Performance comparisons with the centralized training in different asynchrony levels, in a 5-user scenario.*

IV. Numerical Experiments

We consider a 5-user network with independent channel coefficients $h_{i,j} \sim \mathcal{N}(0,1)\ \forall i,j \in [5]$. The resource allocation task considered in the simulations is power allocation, i.e., $p_i = \phi_i(\boldsymbol{s}_i, \boldsymbol{\theta}_i^\star)$, with the user data rate as its utility. In the experiments, we set $p_{max} = 10, \sigma_n^2 = 0.5, \mu = 1, \gamma_0 = 1$, $\tilde{\gamma} = 0.5$, and $B = 100$ with $B_1 = B_2 = 50$.

Each user has a DNN with 3 hidden layers and $\{4, 30, 30, 1\}$ neurons respectively. As the input of the DNN, each user i measures the vector $\boldsymbol{s}_i = (\mathcal{R}, \mathcal{T}, \mathcal{G}, \mathcal{P})$, including the received noise plus interference power from all the other users (the denominator in (1)), the transmitted interference power $|h_{ji}|^2 p_i$ to each user j separately, the single direct channel $|h_{ii}|^2$, and the previous transmit power p_i, respectively. These inputs are among the inputs that [6], [20], [23] have used for their policies. The transmitted interference power can be available at each user by message passing with the most affected neighbors. The power range $[0, p_i^{max}]$ is enforced by the *sigmoid* activation function at the DNN output layer. We consider two scenarios for the proposed method, namely the first scenario with a *fully connected network* and the second scenario with a *connected network* where the network graph is randomly generated with a maximum size of 2 for $\mathcal{N}_i,\ i \in [N]$.

The proposed approach is compared against two benchmarks, namely the centralized training approach which uses first-order information i.e., the exact gradient, of the reward function (2) to perform updates by the Adam [24] optimizer, and FL-based approach [20] where the users exchange their parameters $\boldsymbol{\theta}_i$ with a server every I iterations to reach a consensus $\boldsymbol{\theta}_1 = \cdots = \boldsymbol{\theta}_N$. We report the asynchrony level by the maximum time difference between the users as a percentage of the batch size B. Figs. 2 and 3 show the performance of the proposed decentralized approach compared to the two benchmarks versus the number of channel instances (samples). The proposed approach reaches the same performance with a large reduction in communication overhead. Fig. 4 shows that smaller batch sizes are preferable as there are more updates in step 7 of Algorithm 1 in the same time period compared to when larger batch size is used. Increasing the asynchrony level, as in Fig. 5, decreases the performance, yet, the proposed approach shows reasonable robustness against asynchrony. To evaluate if users have different policies after convergence, we perform the proposed training when the users have different path loss values. In the inference stage, the achieved sum rate (SR) is then evaluated when either each user uses its own policy or the trained policies are randomly assigned to the users. As apparent from Table I, each user policy has learned the unique communication status of the user, which indeed results in a higher SR.

The proposed approach is also evaluated for larger communication networks when the users are synchronous. With results provided in Table II, it is evident that the proposed decentralized approach can perform the training well also when the communication network is larger.

V. Conclusion

In this paper, we have presented a training approach that can be combined with the existing decentralized DL based RA methods, to train their policies in a decentralized asynchronous and model-free manner. With the proposed approach, the users cooperatively train their policies when they are linked by a fully or partially connected graph. The studied asynchrony addresses the misalignment between the users in their updates when they all have the same update frequency. Other sources

Table I: *Performance comparisons with four random policy assignments in different asynchrony levels when $N = 5$.*

	Sum Rate (bits/s/Hz)				
async.	**no rnd.**	**rnd. 1**	**rnd. 2**	**rnd. 3**	**rnd. 4**
0%	4.34	1.81	1.12	2.22	2.05
10%	4.63	3.67	2.29	2.04	2.86
25%	3.96	3.71	2.98	3.17	3.47
50%	3.82	2.06	1.87	3.26	3.21

Table II: *Performance comparisons after convergence.*

Sum Rate (bits/s/Hz)		
# of users	**proposed**	**centralized Adam**
5	4.95	5.12
10	5.31	5.35
15	5.63	5.71

of asynchrony may be interesting to investigate as well. A theoretical study of convergence behavior is also a direction for future research.

REFERENCES

[1] V. P. Mhatre, K. Papagiannaki, and F. Baccelli, "Interference mitigation through power control in high density 802.11 WLANs," in *26th IEEE International Conference on Computer Communications (INFOCOM)*, 2007, pp. 535–543.

[2] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards Optimal Power Control via Ensembling Deep Neural Networks," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1760–1776, 2020.

[3] 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.

[4] D. S. Kalogerias, M. Eisen, G. J. Pappas, and A. Ribeiro, "Model-Free Learning of Optimal Ergodic Policies in Wireless Systems," *IEEE Transactions on Signal Processing*, vol. 68, pp. 6272–6286, 2020.

[5] P. Behmandpoor, J. Verdyck, and M. Moonen, "Deep learning-based cross-layer resource allocation for wired communication systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 4120–4124.

[6] 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.

[7] C. Guo, Z. Li, L. Liang, and G. Y. Li, "Reinforcement Learning Based Power Control for Reliable Wireless Transmission," *arXiv preprint arXiv:2202.06345*, 2022.

[8] P. Behmandpoor, P. Patrinos, and M. Moonen, "Learning-Based Resource Allocation with Dynamic Data Rate Constraints," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 4088–4092.

[9] M. Eisen and A. Ribeiro, "Optimal Wireless Resource Allocation With Random Edge Graph Neural Networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2977–2991, 2020.

[10] N. NaderiAlizadeh, M. Eisen, and A. Ribeiro, "Learning resilient radio resource management policies with graph neural networks," *IEEE Transactions on Signal Processing*, vol. 71, pp. 995–1009, 2023, publisher: IEEE.

[11] ——, "State-augmented learnable algorithms for resource management in wireless networks," *IEEE Transactions on Signal Processing*, vol. 70, pp. 5898–5912, 2022.

[12] M. Zecchin, D. Gesbert, and M. Kountouris, "Team Deep Mixture of Experts for Distributed Power Control," in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.

[13] Z. Wang, M. Eisen, and A. Ribeiro, "Unsupervised learning for asynchronous resource allocation in ad-hoc wireless networks," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8143–8147.

[14] W. Cui and W. Yu, "Uncertainty injection: A deep learning method for robust optimization," *IEEE Transactions on Wireless Communications*, 2023.

[15] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep Reinforcement Learning Based Resource Allocation for V2V Communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[16] M. Eisen, K. Gatsis, G. J. Pappas, and A. Ribeiro, "Learning in wireless control systems over nonstationary channels," *IEEE Transactions on Signal Processing*, vol. 67, no. 5, pp. 1123–1137, 2018.

[17] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep Learning for Radio Resource Allocation With Diverse Quality-of-Service Requirements in 5G," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2309–2324, 2021.

[18] R. Nassif, S. Vlaski, C. Richard, and A. H. Sayed, "Learning Over Multitask Graphs—Part I: Stability Analysis," *IEEE Open Journal of Signal Processing*, vol. 1, pp. 28–45, 2020.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[20] P. Behmandpoor, P. Patrinos, and M. Moonen, "Federated Learning Based Resource Allocation for Wireless Communication Networks," in *EUSIPCO 30th European Signal Processing Conference*, 2022, pp. 1656–1660.

[21] M. Yan, B. Chen, G. Feng, and S. Qin, "Federated Cooperation and Augmentation for Power Allocation in Decentralized Wireless Networks," *IEEE Access*, vol. 8, pp. 48 088–48 100, 2020.

[22] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.

[23] W. Cui, K. Shen, and W. Yu, "Spatial Deep Learning for Wireless Scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1248–1261, 2019.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.