Federated Learning Based Resource Allocation for Wireless 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*—In this paper we introduce federated learning (FL) based resource allocation (RA) for wireless communication networks, where users cooperatively train a RA policy in a distributed scenario. The RA policy for each user is represented by a local deep neural network (DNN), which has the same structure for all users. Each DNN monitors local measurements and outputs a power allocation to the user. The proposed approach is model-free; each user is responsible for training its own DNN to maximize the sum rate (SR) and communicates with the server to aggregate its local DNN with other DNNs. More importantly, each user needs to probe only its own data rate as a distributed reward function and communications with the server once in a while. Simulations show that the proposed approach enables conventional deep learning (DL) based RA methods to not only use their policy in a distributed scenario, but also to (re)train their policy in time-varying environments in a model-free distributed manner without needing a computationally complex server.**

***Index Terms*—Deep neural network, federated learning, distributed reward, wireless communication, resource allocation.**

I. Introduction and motivation

Resource allocation (RA) in wireless and wired communication networks is a challenging task as the underlying optimization problem is nonconvex and in some communication scenarios, large scale [1]–[4].

Recently, to speed up the corresponding optimization in RA, deep learning (DL) based methods have been proposed to exploit the advantages of deep neural networks (DNNs) [5]–[8]. These methods aim to train a DNN, as a function approximator, to approximate an optimum policy that relates optimum resources, e.g. transmit power, to parameters of the problem that are time-varying, e.g. channel coefficients. Unlike conventional iterative optimization-based RA, DL based RA is non-iterative and in closed form, speeding up the process with a simple implementation [6], [9]–[13].

The proposed DL based RA methods in the literature require retraining of their DNN(s) once in a while as the parameters of the communication networks, e.g. the channel distribution, are constantly changing [14], [15]. In this scenario, model-free retraining [11] is motivated to incorporate all real-time drifts from the initial model. In the model-free training, gradient-free optimization is employed by measuring the reward function, rather than representing the reward function by a model and calculating its gradient vector (cf. Section II-C).

Model-free (re)training, however, is only studied in a centralized manner where the server trains a centralized policy and needs to receive all the necessary time-varying communication network parameters, e.g. channel coefficients, over the users to form a centralized reward function, e.g. sum rate (SR) of users. Moreover, the server performing such a training task needs considerable computational power, which is not always available.

In response to the growth of computational capability at the edge, federated learning (FL) has been introduced by [16] and developed as an active field of research in distributed optimization and training [17]. Hence, a natural question that comes to mind is whether *the training in the current DL based RA methods can be done in a model-free distributed manner using the concepts of FL, avoiding the requirement of a computationally complex server*. In this paper, a FL based RA method is proposed for the training phase which lifts the mentioned limitations.

II. Preliminaries

A. System Model

Suppose there are N users, each consisting of a transmitter and a receiver, communicating with each other in a wireless communication network. The transmitter of user i transmits data with power p_i, the ith element of the power vector $\boldsymbol{p}$. Also, the channel between the transmitter of user i and the receiver of user j is denoted by h_{ji} which is an element of channel matrix $\boldsymbol{H}$ in its jth row and ith column. Moreover, we assume i.i.d. additive Gaussian noise with variance σ^2 at the receivers. In this setting, the achievable data rate of user i can be written as,

$$R_i(\boldsymbol{H}, \boldsymbol{p}) = \log_2(1 + \frac{|h_{ii}|^2 p_i}{\sigma^2 + \sum_{j \neq i} |h_{ij}|^2 p_j}). \qquad (1)$$

The objective in this paper is to train a DNN as a RA policy denoted by $\phi(., \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^m$, such that the sum rate (SR) of users are maximized at each time instant. Having trained the RA policy once, each user uses the policy with the optimal parameter $\boldsymbol{\theta}^\star$ to find its optimal transmit power simply as $p_i^\star = \phi(\boldsymbol{s}_i, \boldsymbol{\theta}^\star)$. The vector $\boldsymbol{s}_i$ contains the local measurements, e.g. recieved interference power, performed by the user i to feed the policy. One example of such inputs

This research work was carried out at the ESAT Laboratory of KU Leuven, in the frame of Research Project FWO nr. G.0B1818N 'Real-time adaptive cross-layer dynamic spectrum management for fifth generation broadband copper access networks' and Fonds de la Recherche Scientifique - FNRS and Fonds voor Wetenschappelijk Onderzoek - Vlaanderen EOS Project no 30452698 '(MUSE-WINET) MUlti-SErvice WIreless NETwork'. The scientific responsibility is assumed by its authors. (emails: {pourya.behmandpoor, panos.patrinos, marc.moonen}@esat.kuleuven.be)

Algorithm 1: Federated averaging (FedAve [16])

Initialize: $\boldsymbol{\theta}_i^0 = \boldsymbol{\theta}^0 \in \boldsymbol{\Theta}\ \forall i \in [N]$, set communication interval $I > 0$, set step size sequence $\{\gamma^k\} \subseteq \mathbb{R}_{++}$

for $k = 0, 1, 2, \ldots$ **do**

user $i \in [N]$ in parallel with others performs:

if *k is a multiple of I,* **then**

send $\boldsymbol{\theta}_i^k$ to the server

update $\boldsymbol{\theta}_i^k \leftarrow \bar{\boldsymbol{\theta}}^k$ with $\bar{\boldsymbol{\theta}}^k$ from the server

end

local update: $\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \gamma^k N \nabla_{\theta_i} f(\boldsymbol{\theta}_1^k, \ldots, \boldsymbol{\theta}_N^k)$
$= \boldsymbol{\theta}_i^k + \gamma^k \nabla f_i(\boldsymbol{\theta}_i^k)$

end

is provided in the simulation section. Using a RA policy, such as a DNN, speeds up the RA compared to the conventional RA methods that require performing an optimization task at each time instant. The mentioned parameterized policy ϕ can be trained as follows,

$$\begin{aligned} \boldsymbol{\theta}^\star = &\operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \mathbb{E}_{H\sim\mathcal{D}}\{R_i(\boldsymbol{H}, \boldsymbol{\phi})\} \\ &s.t.\ \ 0 \le \phi(\boldsymbol{s}_i, \boldsymbol{\theta}) \le p_{max}, \quad \forall i \in [N], \end{aligned} \tag{2}$$

where $\boldsymbol{\phi} = [\phi(\boldsymbol{s}_1, \boldsymbol{\theta}), \ldots, \phi(\boldsymbol{s}_N, \boldsymbol{\theta})]^T$, p_{max} is the transmit power constraint, $[N] := \{1, \ldots, N\}$, and $\mathcal{D}$ is the distribution of the channel during the training. Note that the power constraints can be easily met by the structure of the DNN, e.g. using the *sigmoid* activation function at the output layer.

B. Federated Learning

FL has been introduced by [16] and developed later by many researchers in different DL based applications, e.g. in communication networks [18]. The main objective in FL is to optimize the following finite sum of functions in a distributed manner while the communication overhead and user privacy are taken into account:

$$\operatorname{argmax}_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \tfrac{1}{N} \textstyle\sum_{i=1}^{N} f_i(\boldsymbol{\theta}), \tag{3}$$

where there are N users (agents), each with a local reward function f_i. This problem can be cast to the following consensus optimization problem,

$$\begin{aligned} \operatorname{argmax}_{\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_N} f(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N) = \tfrac{1}{N} \textstyle\sum_{i=1}^{N} f_i(\boldsymbol{\theta}_i) \\ s.t. \quad \boldsymbol{\theta}_1 = \boldsymbol{\theta}_2 = \cdots = \boldsymbol{\theta}_N \end{aligned} \tag{4}$$

with local parameters $\boldsymbol{\theta}_i$ and local reward functions $f_i(\boldsymbol{\theta}_i)$. One of the first and the most popular proposed methods in FL called FedAve [16] uses stochastic gradient ascent (SGA). The steps of this method are summarized in **Algorithm 1**.

Note that every I iteration, the server aggregates the parameters as

$$\bar{\boldsymbol{\theta}}^k = \tfrac{1}{N} \textstyle\sum_{i=1}^{N} \boldsymbol{\theta}_i^k \tag{5}$$

Algorithm 2: Gradient-free min. oracle (cf. [21])

Input: f, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, $\mu > 0$, $L \ge 1$
Output: $g(\boldsymbol{\theta}) := \hat{\nabla} f^\mu(\boldsymbol{\theta})$

1 generate $\boldsymbol{u}^l \in \mathbb{R}^m\ \forall l \in [L]$ randomly drawn from the distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, such that $\boldsymbol{\theta} + \mu \boldsymbol{u}^l \in \boldsymbol{\Theta}$

2 evaluate $f(\boldsymbol{\theta})$, $f(\boldsymbol{\theta} + \mu \boldsymbol{u}^l)\ \forall l \in [L]$

3 calculate $g(\boldsymbol{\theta}) = \frac{1}{L} \sum_{l=1}^{L} \frac{f(\boldsymbol{\theta}+\mu\boldsymbol{u}^l) - f(\boldsymbol{\theta})}{\mu} \boldsymbol{u}^l$

to guarantee the equality constraint in (4). The server receives all the local parameters $\boldsymbol{\theta}_i^k$ from the users and sends back the aggregation $\bar{\boldsymbol{\theta}}^k$ to the users. Hence, the parameter $\bar{\boldsymbol{\theta}}^k$ in the server is updated every I iterations as,

$$\bar{\boldsymbol{\theta}}^k = \begin{cases} \bar{\boldsymbol{\theta}}^{k-1}, & \text{otherwise} \\ \bar{\boldsymbol{\theta}}^{k-I} + \frac{1}{N} \sum_{t=0}^{I-1} \sum_{i=1}^{N} \gamma^{k-I+t} \nabla f_i(\boldsymbol{\theta}_i^{k-I+t}), \\ \qquad\qquad \text{if } k \text{ is a multiple of } I \end{cases} \tag{6}$$

with the iteration counter k as the superscript and the step size $\gamma^k = \alpha/(1+k)^\beta$, with constants $\alpha > 0, \beta > 0$, as a diminishing step size is needed in the SGA based methods to converge [19]. Note that in the case of $I = 1$, we would recover the conventional (mini-batch) SGA with $\boldsymbol{\theta}_i^k = \bar{\boldsymbol{\theta}}^k$ due to the aggregation step (5) and we have,

$$\bar{\boldsymbol{\theta}}^{k+1} = \bar{\boldsymbol{\theta}}^k + \tfrac{\gamma^k}{N} \textstyle\sum_{i=1}^{N} \nabla f_i(\bar{\boldsymbol{\theta}}^k) \tag{7}$$

However, $I = 1$ incurs a high communication overhead.

C. Gradient-Free Minimization

We are interested in only measuring the reward function, rather than representing the reward function by a model and estimating the parameters involved. As an advantage, we can bypass the estimation of parameters, e.g. channel matrix and noise power, and be more robust against the uncertainties in modeling, e.g. nonlinearities [11]. Random optimization approach [20], [21] is a popular zero-order optimization approach that only requires measurements of the reward function rather than computing the reward gradient vector. In [11] this approach is used in RA as a model-free method.

Let a so-called smoothed function f^μ as,

$$f^\mu(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{u}}\{f(\boldsymbol{\theta} + \mu \boldsymbol{u})\}, \tag{8}$$

where $\mu > 0$ is a sufficiently small constant and $\boldsymbol{u}$ is a random vector. The gradient-free minimization method performs SGA using an estimation of the gradient vector $g(\boldsymbol{\theta}) := \hat{\nabla} f^\mu(\boldsymbol{\theta})$ as its oracle to converge to a locally optimal point of function f up to the precision $\epsilon(\mu)$. The oracle g is calculated using **Algorithm 2** with L number of repeats in the finite difference calculation, and $\boldsymbol{\Sigma}$ as the identity matrix.

III. Federated Learning Based Resource Allocation

In this section, using the concepts of FL, the objective is to propose a policy training approach for RA that offloads

the policy training task from the server. Also, to limit the communication overhead, the users are not allowed to share their local measurement vectors $\boldsymbol{s}_i$ with each other and they only communicate with the server once in a while. Moreover, to avoid probing delays in model-free updates, the users enjoy distributed reward function, i.e. each user only tracks its own data rate rather than the SR.

A. FL Based Reformulation

By comparing equations (2) and (4), one can formulate:

$$\begin{aligned} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i}) &= \mathbb{E}_{\boldsymbol{H}\sim\mathcal{D}}\{R_i(\boldsymbol{H}, \{\phi(\boldsymbol{s}_i, \boldsymbol{\theta}_i), \boldsymbol{\phi}_{\setminus i}\})\} \\ f(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N) &= \tfrac{1}{N}\textstyle\sum_{i=1}^{N} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i}) \end{aligned} \tag{9}$$

where $\boldsymbol{\phi}_{\setminus i}$ is a vector containing $\phi(\boldsymbol{s}_j, \boldsymbol{\theta}_j)$ for $j \in [N] \setminus i$, where j can take any value in the set $[N]$ except i . Note that due to the interference, each data rate R_i depends not only on $\phi(\boldsymbol{s}_i, \boldsymbol{\theta}_i)$, but also on $\phi(\boldsymbol{s}_j, \boldsymbol{\theta}_j)$. Hence, in this application, we cannot completely disjoin parameters $\boldsymbol{\theta}_i$ like in the consensus formulation (4), and the local functions are in the form of $f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i})$ rather than $f_i(\boldsymbol{\theta}_i)$.

By the formulation (9), each user i performs its local update as (cf. Algorithm 1),

$$\begin{aligned} &N\nabla_{\theta_i} f(\boldsymbol{\theta}_1^k, \ldots, \boldsymbol{\theta}_N^k) \\ &\qquad = \nabla_{\boldsymbol{\theta}_i} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i}) + \sum_{j=1, j\neq i}^{N} \nabla_{\boldsymbol{\theta}_i} f_j(\boldsymbol{\theta}_j, \boldsymbol{\theta}_{\setminus j}). \end{aligned} \tag{10}$$

However, user i does not have access to the local reward functions $f_j(\boldsymbol{\theta}_j, \boldsymbol{\theta}_{\setminus j})$, since to avoid communication overhead, it is not allowed to get the parameters of these functions, e.g. $\boldsymbol{s}_j$ and corresponding channel coefficients.

Since all the gradients w.r.t. the local parameters $\boldsymbol{\theta}_i$ are aggregated often every I iterations by the server as in (5) and (6), we have:

$$\begin{aligned} &N\sum_{i=1}^{N} \nabla_{\theta_i} f(\boldsymbol{\theta}_1^k, \ldots, \boldsymbol{\theta}_N^k) \\ &= \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}_i} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i}) + \sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} \nabla_{\boldsymbol{\theta}_i} f_j(\boldsymbol{\theta}_j, \boldsymbol{\theta}_{\setminus j}) \\ &= \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}_i} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i}) + \sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} \nabla_{\boldsymbol{\theta}_j} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i}) \\ &= \sum_{i=1}^{N} \{\bar{\boldsymbol{y}}_i + \tilde{\boldsymbol{y}}_i\}. \end{aligned} \tag{11}$$

Hence each user can perform its local update as,

$$\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \gamma^k(\bar{\boldsymbol{y}}_i + \tilde{\boldsymbol{y}}_i), \tag{12}$$

using only its own local reward function f_i. However, the term $\tilde{\boldsymbol{y}}_i$ is not easily available in user i due to the need for the parameters $\boldsymbol{\theta}_j$. The next section proposes a remedy for this issue.

B. Distributed Reward

Note that the policy parameters $\boldsymbol{\theta}_j$ with $j \neq i$ are not available in user i to compute the corresponding derivative $\tilde{\boldsymbol{y}}_i$ for (12). In the conventional DL based RA methods, a *centralized reward function*, e.g. the SR of users, is available at the server, making the implementation of (12) possible. However, in our proposed FL approach, we are interested in having *distributed reward functions* so that each user does not need other users' reward functions and local parameters to perform its local updates. In this way, the communication overhead and possible delays during the parameter sharing are avoided. Furthermore, it is interesting to have a model-free approach during the training to enjoy the advantages mentioned in Section II.C.

To do so, the gradients $\nabla_{\boldsymbol{\theta}_j} f_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{\setminus i})$ in the $\tilde{\boldsymbol{y}}_i$ part of (12) can be estimated at user i by the gradient-free minimization oracle in Algorithm 2 as,

$$\begin{aligned} \hat{\nabla}_{\boldsymbol{\theta}_j} f_i^{\mu}(\boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{\setminus i}^k) &:= g_i^j(\boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{\setminus i}^k) \\ &= \frac{1}{L}\sum_{l=1}^{L} \frac{f_i(\boldsymbol{\theta}_j^k + \mu\boldsymbol{u}^l, \boldsymbol{\theta}_{\setminus j}^k) - f_i(\boldsymbol{\theta}_j^k, \boldsymbol{\theta}_{\setminus j}^k)}{\mu}\boldsymbol{u}^l \end{aligned} \tag{13}$$

once user j changes its parameter $\boldsymbol{\theta}_j^k$. Here, $\hat{\nabla} f_i^{\mu}$ is the estimated gradient of f_i^{μ}, the so-called smoothed version of the function f_i –cf. Section II.C. Note that the values $f_i(\boldsymbol{\theta}_j^k, \boldsymbol{\theta}_{\setminus j}^k)$ and $f_i(\boldsymbol{\theta}_j^k + \mu\boldsymbol{u}_j^l, \boldsymbol{\theta}_{\setminus j}^k)$ can be measured by user i, by probing only its own data rate, respectively before and after the time instant when user j changes its local parameter $\boldsymbol{\theta}_j$. A batchsize B is considered to estimate the expectation in (9) by averaging. The random vectors $\boldsymbol{u}^l$ can be known in all users, using a common known distribution and a common random seed. Hence, no communication is needed to exchange the local parameters between the users, motivating us to call $\bar{\boldsymbol{y}}_i + \tilde{\boldsymbol{y}}_i$ in the local update rule of (12) a *distributed reward function*.

In this approach, the users sequentially update their parameters in a synchronized manner as follows. Each user i updates its parameter either by $\bar{\boldsymbol{y}}_i$ in (12), i.e. by measuring the gradient of its local function w.r.t. its local parameter $\boldsymbol{\theta}_i^k$ as,

$$\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \gamma^k g_i^i(\boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{\setminus i}^k)$$

or whenever another user j updates its parameter $\boldsymbol{\theta}_j^k$, the user i in parallel records the changes of its own local reward function $f_i(\boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{\setminus i}^k)$ caused by the change in the parameter $\boldsymbol{\theta}_j^k$ as,

$$\boldsymbol{\theta}_i^{l_i} \leftarrow \boldsymbol{\theta}_i^{l_i} + \gamma^{l_i} g_i^j(\boldsymbol{\theta}_i^{l_i}, \boldsymbol{\theta}_{\setminus i}^{l_i})$$

where $l_i \in \{k, k+1\}$ is the current time index of user i. Note that the time index moves one step forward only when each user updates its parameter by $\bar{\boldsymbol{y}}_i$. If user i has already updated its parameter by $\bar{\boldsymbol{y}}_i$, $l_i = k+1$ otherwise $l_i = k$. Also the server aggregates the parameters every I iterations by,

$$\bar{\boldsymbol{\theta}} = \tfrac{1}{N}\textstyle\sum_{\ell=1}^{N} \boldsymbol{\theta}_{\ell}^{l_\ell}. \tag{14}$$

The proposed approach is summarized in **Algorithm 3**.

Algorithm 3: Proposed FL based RA approach

Initialize: $\boldsymbol{\theta}_i^0 = \boldsymbol{\theta}^0 \; \forall i \in [N]$, $\alpha > 0$, $\beta > 0$ for $\gamma^k = \frac{\alpha}{(1+k)^\beta}$, $L \geq 1$, $I > 1$ and batchsize $B \geq 1$

Output: $\boldsymbol{\theta}^K$

for $k = 0, 1, \ldots, K$ **do**

  $\boldsymbol{u}^l \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \; \forall l \in [L]$ in all users

  $l_i = k \quad \forall i \in [N]$

  for $i = 1, 2, \ldots, N$ sequentially **do**

    user i

      estimates: $g_i^i(\boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{\setminus i}^k)$ by (13)

      performs: $\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k + \gamma^k g_i^i(\boldsymbol{\theta}_i^k, \boldsymbol{\theta}_{\setminus i}^k)$

      $l_i = k + 1$

    user j, $\forall j \in [N] \setminus i$

    in parallel and synchronous with user i

      estimates: $g_j^i(\boldsymbol{\theta}_j^{l_j}, \boldsymbol{\theta}_{\setminus j}^{l_j})$ by (13) (15)

      performs: $\boldsymbol{\theta}_j^{l_j} \leftarrow \boldsymbol{\theta}_j^{l_j} + \gamma^{l_j} g_j^i(\boldsymbol{\theta}_j^{l_j}, \boldsymbol{\theta}_{\setminus j}^{l_j})$

    if $(kN + i)$ *is a multiple of* I, **then**

      all users send $\boldsymbol{\theta}_\ell^{l_\ell} \; \forall \ell \in [N]$ to the server

      all users update $\boldsymbol{\theta}_\ell^{l_\ell} \leftarrow \bar{\boldsymbol{\theta}} \; \forall \ell \in [N]$, with $\bar{\boldsymbol{\theta}}$ in (14) from the server

    end

  end

end

Note that the steps (15) in Algorithm 3 are unique steps in RA compared to vanilla FL methods. The reason is that in RA the local reward functions cannot disjoin to form the consensus optimization problem of (4) due to the interference between the users.

IV. Numerical Simulations

We consider a network with $N = 5$ users. The channel matrix $\boldsymbol{H} \in \mathbb{R}^{5\times5}$ is randomly generated with independent $h_{i,j} \sim \mathcal{N}(0,1) \; \forall i,j \in [5]$ elements. In all the simulations, we set $p_{max} = 10, \sigma^2 = 0.5, \mu = 10^{-6}, L = 1, \alpha = 0.75$, $\beta = 0.5$, and $B = 100$.

Each user has a DNN with 3 hidden layers with $\{3, 25, 25, 1\}$ neurons respectively. As the input of the DNN, each user i measures the vector $\boldsymbol{s}_i = \{\mathcal{R}, \mathcal{T}, \mathcal{G}\}$, indicating the received interference by the user i, the transmitted interference by the user i, and the direct channel, respectively. These inputs are among the inputs that [8], [22] have used for their policies. Each user has access to only its own data rate R_i by probing its communication link. Using the DNN, each user controls its transmit power $p_i \in [0, p_{max}]$. The power range $[0, p_{max}]$ is enforced by the *sigmoid* activation function at the output layer. The activation functions of the other layers are given as $relu(x) = \max\{0, x\}$.

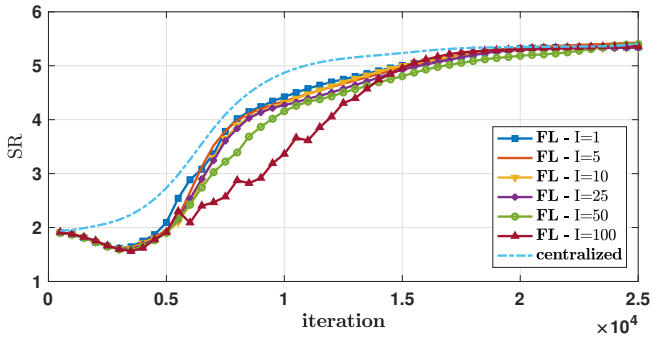

Fig. 1. Comparison of the proposed FL based RA approach with the centralized RA, for different aggregation intervals I.

TABLE I
Performance comparisons of the proposed FL based model-free RA method with $I = 50$ and $I = 100$ with the centralized model-based scenario.

Sum Rate (bits/s/Hz)			
# of users	**FL** ($I = 50$)	**FL** ($I = 100$)	**centralized**
5	5.36	5.35	5.36
10	6.42	6.41	6.42
15	7.00	7.00	7.01

In the first simulation, each user follows the steps outlined in Algorithm 3 to train its DNN. We compare the SR of the users when different aggregation intervals I are considered during the training. We also compare our proposed FL based model-free RA approach with the case the DNN is trained in a centralized model-based scenario with exact gradient vectors ∇f_i and aggregation interval $I = 1$, where the SR of the users is known and used for the training. Fig. 1 shows that the performance of the proposed approach in terms of the SR reaches that of the centralized model-based setting, even when I is as large as $I = 100$, which has a considerable advantage in practice. Note that the convergence is slower due to the aggregation interval $I > 1$ and the noisy gradient estimates g_i^j for all $i, j \in [N]$. In Table 1, the SR is reported after convergence, comparing the proposed FL based RA method with the centralized model-based scenario, for three different numbers of users N. The same convergence behavior as Fig. 1 is noticed for $N = 10$ and $N = 15$.

The communication load for the aggregation step (14) is compared in Fig. 2 for different aggregation intervals I. Note that the large intervals I are favorable in terms of communication overhead-convergence speed trade-off.

In our last simulation, each user trains its policy without any communication with the server, i.e. $I \to \infty$. As it is obvious in Fig. 3, without exchanging parameters $\boldsymbol{\theta}_i$, each user learns to selfishly maximize its transmit power over time, until it reaches the maximum power p_{max}. This behavior is understandable, as each user only has access to its own data rate, and thus maximizes this data rate without noticing to

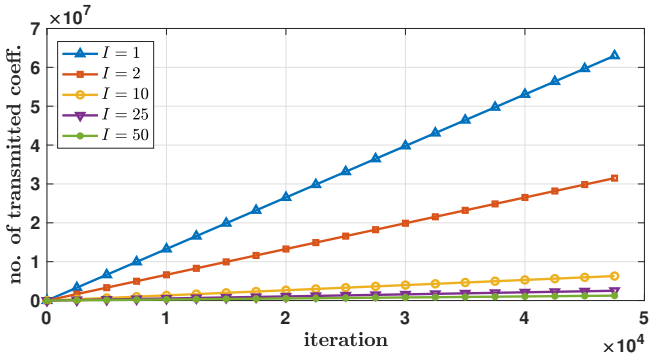

Fig. 2. Comparison of the communication overhead for different aggregation intervals I.

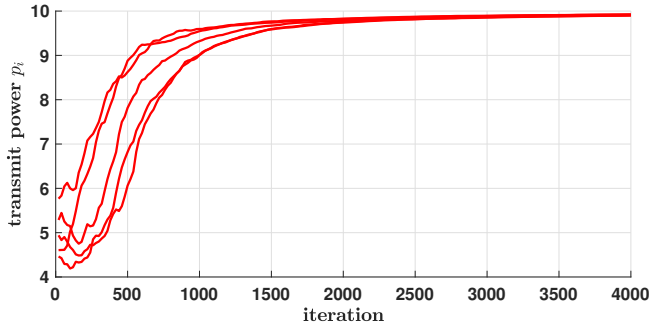

Fig. 3. Transmit power of users when there is no parameter aggregation (14), i.e. $I \to \infty$, during the training.

what extent its transmit power can negatively affect the other users' data rates.

V. Conclusion and Future Works

In this paper, we have proposed a model-free policy (re)training approach based on FL concepts, where the policy is used for RA to maximize SR at each time instant. The training is done in a distributed manner, where the users communicate with a computationally simple server once in a while. More importantly, each user needs to probe only its own data rate, as the distributed reward function, to update its local DNN. Furthermore, the users do not need to share their local measurements with the server during the training. All these features offload the training computational load from the server and guarantee the possibility of real-time model-free policy (re)training in time-varying environments without necessarily requiring a computationally complex server.

Asynchronous extensions in heterogeneous systems, decentralized versions without requiring a server, and convergence studies are topics of future research activities.

References

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

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

[3] M. Chiang, C. W. Tan, D. P. Palomar, D. O'neill, and D. Julian, "Power control by geometric programming," *IEEE transactions on wireless communications*, vol. 6, no. 7, pp. 2640–2651, 2007.

[4] R. Cendrillon, J. Huang, M. Chiang, and M. Moonen, "Autonomous spectrum balancing for digital subscriber lines," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4241–4257, 2007.

[5] F. Zhou, G. Lu, M. Wen, Y.-C. Liang, Z. Chu, and Y. Wang, "Dynamic spectrum management via machine learning: state of the art, taxonomy, challenges, and open research issues," *IEEE Network*, vol. 33, no. 4, pp. 54–62, 2019.

[6] 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, 2019.

[7] 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.

[8] 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, p. 2239, 2019.

[9] M. Eisen, C. Zhang, L. F. Chamon, D. D. Lee, and A. Ribeiro, "Learning optimal resource allocations in wireless systems," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2775–2790, 2019.

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

[11] 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.

[12] P. Behmandpoor, P. Patrinos, and M. Moonen, "Learning-based resource allocation with dynamic data rate constraints," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4088–4092.

[13] M. Zecchin, D. Gesbert, and M. Kountouris, "Team deep mixture of experts for distributed power control," in *21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.

[14] 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.

[15] M. Kim, P. de Kerret, and D. Gesbert, "Learning to cooperate in decentralized wireless networks," in *52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2018, pp. 281–285.

[16] 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, p. 1273.

[17] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[18] A. M. Elbir and S. Coleri, "Federated learning for channel estimation in conventional and irs-assisted massive mimo," *arXiv preprint arXiv:2008.10846*, 2020.

[19] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5693–5700.

[20] J. Matyas *et al.*, "Random optimization," *Automation and Remote control*, vol. 26, no. 2, pp. 246–253, 1965.

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

[22] 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.