Federated Deep Reinforcement Learning for the Distributed Control of NextG Wireless Networks

Aakash Agarwal
ECE 1910201
Internation Institute of Information Technology
Naya Raipur

Nikhil Vishwakarma
ECE 1910224
Internation Institute of Information Technology
Naya Raipur

Piyush Rawat
ECE 191010227
Internation Institute of Information Technology
Naya Raipur

***Abstract*—Next Generation (NextG) networks are expected to support demanding tactile internet applications such as augmented reality and connected autonomous vehicles. Whereas recent innovations bring the promise of larger link capacity, their sensitivity to the environment and erratic performance defy traditional model-based control rationales. Zero-touch datadriven approaches can improve the ability of the network to adapt to the current operating conditions. Tools such as reinforcement learning (RL) algorithms can build optimal control policy solely based on a history of observations. Specifically, deep RL (DRL), which uses a deep neural network (DNN) as a predictor, has been shown to achieve good performance even in complex environments and with high dimensional inputs. However, the training of DRL models require a large amount of data, which may limit its adaptability to ever-evolving statistics of the underlying environment. Moreover, wireless networks are inherently distributed systems, where centralized DKL approaches would require excessive data exchange, while fully distributed approaches may result in slower convergence rates challenges, we propose a federated learning, (FL) approach to DRL, which propose a federated learning (FL) approach to DKL, which we refer to federated DRL (F-DRL), where base sharing models' weights rather than training data. We evaluate two distinct versions of F-DRL, value and policy based, and show the superior performance they achieve compared to distributed and centralized DRL.**

***Index Terms*—reinforcement learning, Federated Learning, Power control, Multi agent reinforcement learning, Wireless networks, Resource allocation.**

I. INTRODUCTION

On the one hand, next generation (NextG) networks are expected to support a wide range of essential and demanding real-time services such as augmented reality, connected autonomous vehicles and in-network computing that require coherent performance. On the other hand, recent advancements at the physical layers such as millimeter Wave (mmW) communications, while empowering the network with increased capacity, make its temporal behavior more erratic and convoluted. The NextG network environment, then, presents inherent control challenges that defy traditional model-based control approaches. To address these daunting challenges, the zero-touch network paradigm utilizes machine learning to eliminate the need for human-based design, and enable fast data-driven adaptation to different operating conditions.

This work was partially supported by the NSF grants MLWiNS-2003237 and CNS-2134567. Power and bandwidth allocation is one of the fundamental problems in wireless networks. While many optimization frameworks have been proposed in this domain [1], by directly utilizing the data originated by the system, data-driven methodologies have the potential to improve the adaptability of the network to environmental and traffic conditions and, ultimately, improve performance. Moreover, they are inherently more robust compared to model-based ones, as the latter class may suffer from model mismatch in real-world settings.

Among data driven algorithms, deep reinforcement learning (DRL) has achieved state-of-the-art performance in high dimensional complex control problems [2]. In DRL frameworks, the agent iteratively interacts with the environment to learn the optimal control policy. Additionally, DRL is often much faster in selecting the optimal action compared to conventional optimization methods which usually requires iterative computation and matrix inversion.

Most of the previous works in this domain proposed either fully centralized [3], [4] or distributed [5], [6] DRL algorithms to solve the resource allocation problem. In the former case, training and decision making is centralized, where all BSs send all state information at every time step to the server. In the latter case, each BS independently trains and executes the model without sharing information with other BSs.

In this paper, we propose a federated implementation of DRL algorithms which takes the advantages of both distributed and centralized solutions. Federated Learning (FL) [7] is a family of machine learning problems where many clients ($e.g.$, mobile devices or whole organizations) collaboratively train a DNN model under the orchestration of a central server (e.g., a service provider), while keeping the training procedure decentralized. FL embodies the principles of focused collection and data minimization, and can mitigate many of the systemic privacy risks and costs resulting from traditional, centralized, training of machine learning models. This area has received

considerable recent interest, both from academy and industry. However, so far FL have mainly been applied to supervised learning problems in fields such as NLP and machine vision, and there have been few contributions that used federated learning to train distributed control and DRL models. In this domain, a federated control approach has been proposed in [8] to solve coordination problems among a multitude of agents, edge caching problems [9], [10] and mobile edge computing strategies and offloading [11], [12] and other internet of things (IoT) applications [13].

Here we consider the maximization of the downlink sum rate of a multi-cell network with mobile users. The base stations (BSs) have access to local information and collaborate with each other by sharing their model weights in a FL fashion to quickly train the DNN model at the core of the DRL controller. We will consider different value-based and policy-based DRL algorithms and compare the performance and efficiency of distributed and federated versions of these algorithm. We demonstrate that our F-DRL approach improves the performance in terms of overall network sum rate compared to fully distributed implementations by more than 40%. Our F-DRL approach also reduces communication overhead between BSs and the central server compared to fully centralized solutions and also only need to share the model weights publicly so would protect the local users information privacy in each cells.

II. LITERATURE REVIEW

Recent work proposed the use of DRL algorithms to solve a wide range of optimization problems in wireless communication networks. Power allocation is one the main areas, with several contributions using DRL to find the optimal power [4], [5], [14] – [17]. Some of these contributions use deep Qlearning on discrete power levels [4], [15], [18] and other states of the art DRL algorithms, such as deep deterministic policy gradient (DDPG) [4], [5] and trust region policy optimization (TRPO) [3] on continuous power control in multi cell network scenarios. In addition to power allocation, recent work explored the use of DRL for a wide range of resource management problems, including spectrum allocation [19], joint user association and resource allocation [17] and channel selection [20].

In this paper, we expand on this exciting area by proposing a framework where multiple - federated - DRL agents collaboratively learn a shared predictive model, while all training data remain local to the corresponding device (that is, the BS). The key motivations to integrate FL with DRL are (i) the boosting in training speed compared to fully distributed implementation, and (ii) the reduced amount of data to be shared. Regarding the latter point, we note that centralized DRL approaches to this problem need to exchange data for real-time control, while in our F-DRL frameworks the agents exchange the model weights to update the individual models. Thus, the data exchange has a more relaxed delay constraint and the backbone network's load decreases.

III. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a cellular network, where N different Base Stations (BS) serve K mobile users. We assume that both the BSs and users are equipped with one transmit antenna and each BS is deployed at the cell center. We index the BSs with $n \in \mathcal{N} = \{1, 2, \ldots, N\}$ and the users with $k \in \mathcal{K} = \{1, 2, \ldots, K\}$. We denote the channel gain between the nth BS and k th user in cell j at time slot t with:

$$g^t_{n,j,k} = \left|h^t_{n,j,k}\right|^2 \alpha_{n,j,k},$$

where $h^t_{n,j,k}$ is the small scale fading factor with Rayleigh distributed envelope and $\alpha_{n,j,k}$ is the large-scale fading component, which includes path loss and log-normal shadowing. We model the small-scale Rayleigh fading component according to the Jakes fading model, that is, $h^t_{n,j,k}$ is assumed to be a first-order complex Gauss-Markov process:

$$h^t_{n,j,k} = \rho h^{t-1}_{n,j,k} + \sqrt{1-\rho^2} e^t_{n,j,k}.$$

Here, the innovation process variables $e^t_{n,j,k}$ are identically distributed circularly symmetric complex Gaussian random variables with unit variance, independent from $h^{t-1}_{n,j,k}$. The temporal correlation between two consecutive fading component is

$$\rho = J_0\left(2\pi f_d T_s\right),$$

where $J_0(.)$ is the zeroth-order Bessel function of the first kind, f_d is the maximum Doppler frequency, and T_s is the duration of one time slot. A higher mobility of users leads to a higher Doppler frequency and thus a lower temporal correlation of the channel.

Denoting the transmission power from BS n to its user k at slot t with $p^t_{n,k}$, the downlink signal-to-interference-plus-noise ratio (SINR) of user k in cell n at time slot t is:

$$\gamma_{n,k} = \frac{p^t_{n,k} g^t_{n,j,k}}{I_i + I_o + N_k},$$

where N_k is the noise power at user k and I_i and I_o are the intra-cell and inter-cell interference, respectively:

$$I_i = \sum_{k' \neq k} g^t_{n,n,k} p^t_{n,k'}$$
$$I_o = \sum_{n' \neq n} g^t_{n',n,k} \sum_j p^t_{n',j}.$$

The data rate at user k in cell n, then, is:

$$C_{n,k} = B \log\left(1 + \gamma_{n,k}\right),$$

where B is the bandwidth available to the network.

Our goal is to find the set of downlink transmission powers $p^t_{n,k}$ that maximize the sum rate of the whole network under a constraint on the maximum power. Formally, the optimization problem is:

$$\max_{p^t_{n,k}} \sum_n \sum_k C_{n,k}$$
$$\text{s.t.} \quad 0 \leqslant p^t_{n,k} \leqslant P_{\max} \quad \forall k, n,$$

where $P_{\max}$ is the the maximum transmission power for k th AP and minimum data rate requirement at k th user, respectively.

Clearly, due to the interference terms in the denominator of SINRs, the optimization problem is non-convex and its solution non trivial. Importantly, non-convexity is not the only challenge to overcome to solve the problem. In fact, while iterative algorithms can be developed that achieve good performance, these algorithms require compute-intense operations such as matrix inversion and bisection or singular value decomposition in each iteration, which makes their real-time

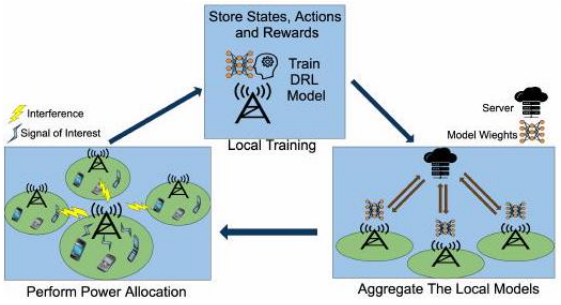

Fig. 1. Fig. 1. Overall view of the network model and F-DRL procedure.

implementation difficult. Additionally, these algorithms need full access to channel state information (CSI) of all the users to derive the optimal solution.

Therefore, the resolution of the optimization problem requires a self-adaptive solution feasible for execution at runtime while achieving good performance having access only to partial observations of the environment. We, then, reformulate the problem as a multi agent RL (MARL) problem, where each access point is an agent which in each time slot determines the transmit power allocated to its associated users. Then, based on the feedback that it receives from the network (which could be a function of rate and powers of other users and neighbor BSs), the BSs adapt their transmit power. Fig. 1 illustrates the framework wropose.

IV. FEDERATED DEEP REINFORCEMENT LEARNING

First, we redefine the problem (8) in a RL setting, where each BS is an agent whose objective is to maximize the sum rate of its own users, while mitigating interference to neighboring cells. Thus, each BS has separate control policies that output the optimal power levels given the current observed state. In the context of DRL, the base stations need to train DNN models whose output is either the Q-values or the control action (defined later). One of the critical issues, then, is training the DNN models as fast as possible to adapt to the current network conditions. In order to speed up training, we propose the federated deep reinforcement learning (FDRL) framework, where the federated agents - the BSs - collaboratively learn a predictive model by sharing their DRL model weights while keeping their users data private.

In the following sections, first we define the RL problem in terms of state space, action space and the reward function corresponding to problem (8). Then, we propose two versions of FDRL: federated deep Q network (FDQN) and federated deep policy gradient algorithm (FDPG) to solve the distributed power control problem.

V. A. RL FORMULATION

RL algorithms are usually set in the context of Markov decision processes (MDP), defined by the 5-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\Pr(s_{t+1} \mid s_t, a_t) \in \mathcal{P}$ are the transition probabilities, $r_t(s_t, a_t) \in \mathcal{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. At each time t, based on the current state s_t, the agent takes an action $a_t \in \mathcal{A}$ and transitions from state s_t to a new state s_{t+1} with probability $\Pr(s_{t+1} \mid s_t, a_t)$, receiving a reward r_t. We define the policy $\pi(s, a)$ as the probability of taking action $a_t = a$ in state $s_t = s$, that is, $\pi(s, a) = \Pr(a_t = a \mid s_t = s)$.

The goal of the RL agent is to learn a policy that maximizes the expected sum of discounted rewards that it receives over the long run, also called return: $R_t = \sum_{i=0}^{\infty} \gamma^{i+t} r_{t+i}$. We define the optimal policy π^* as the policy that maximizes the expected return from a state s :

$$\pi^* = \underset{\pi}{\operatorname{argmax}} E_\pi \{R_0 \mid s_0 = s\}.$$

Here we define the states, actions and reward function for our power control problem.

1) States: The main features that describe the network system are the channel gains between the BSs and users, and the previous transmission power and rate. We assume BSs have only access to information on neighboring cells in the set U_n. The feature set we include in the state of BS n, then, is:
$$\mathcal{S} = \left\{ g^t_{n,j,k}, p^{t-1}_{n,k}, C^{t-1}_{n,k} \right\} \quad \forall j, k \in U_n$$

3) Actions: We use discrete power levels which take values between 0 and $P_{\max}$ defined as
$$\mathcal{A} = \left\{ 0, \frac{P_{\max}}{M-1}, \frac{2P_{\max}}{M-1}, \ldots, P_{\max} \right\}$$
where M is the number of power levels. All agents have the same action space.

4) Reward Function: In order to design the appropriate reward function, we need to estimate the progress of the nth BS toward the goals of the optimization problem (8). To this aim, the reward function considers both the sum-rate of the agent's cell and the interference generated to the neighbor cells U_n. We, then, define the reward function as:
$$r_t = \sum_k C^t_{n,k} + \beta \sum_{n' \in U_n} \sum_k C^t_{n',k}$$
where here β is a parameter controlling the tradeoff between sum-rate and interference. If β is set to 0 , then the agent ignores the performance degradation caused to other cells

(which in turn do the same). As β increases, the BSs take a conservative stance, that is, they privilege interference reduction with respect to their own achieved sum-rate.

VI. B. Federated Learning Formulation

As described later in this section, DRL algorithms use DNNs to produce either Q-values or probability of taking an actions to optimize the return. The canonical federated learning problem involves learning a single, global statistical model (in our case a DNN) from data produced by a multitude of devices. In our framework, we learn this model under the constraint that in each of the BSs state information is stored and processed locally, with only intermediate model updates being communicated periodically to a central server. The goal of the training is to minimize the following objective function:

$$\min_{\theta} F(\theta) = \sum_{i=1}^{N} w_i F_i\left(\theta_i\right),$$

where $F(\theta)$ and θ are the global function and global model weights to be optimized and F_i and θ_i are the local loss function and local model weights at BSi, and w_i is the contribution of cell i to whole network in terms of data size which is equal to $w_i = \frac{k_i}{\sum_{i=1}^{N} k_i}$ where k_i is the number of users in cell i

VII. Federated Deep Q Network

Formally, value-based RL algorithms use an action-value function $Q(s,a)$ to estimate an expected return starting from state s when take action a :

$$\begin{aligned} Q_\pi\left(s_t,a\right) &= E_\pi\left\{\sum_{k=1}^{\infty}\gamma^{k-1} r_{t+k-1} \mid s_t,a\right\} \\ &= E_{s_{t+1},a}\left\{r_t + \gamma Q_\pi\left(s_{t+1},a\right) \mid s_t,a_t\right\} \end{aligned}$$

The optimal action-value function $Q^*\left(s_t,a\right)$ is the cheat sheet sought by the RL agent, which is defined as the maximum expectation of the cumulative discounted return from state s_t :

$$Q^*\left(s_t,a\right) = E_{s_{t+1}}\left\{r_t + \gamma \max_a Q^*\left(s_t,a\right) \mid s_t,a\right\}.$$

In DRL, a function approximation technique (a DNN in the considered case) is used to learn a parametrized value function $Q(s,a;\theta)$ in order to approximate the optimal Q-values. The one-step look ahead $r_t + \gamma \max_a Q\left(s_{t+1},a;\theta_q\right)$ is the target to obtain $Q\left(s_t,a;\theta_q\right)$. Therefore the function $Q\left(s_t,a,\theta_q\right)$ is determined by the parameters θ_q. The selection of a good action relies on accurate action-value estimation, and thus DQN attempts to find the optimal parameters θ_q^* to minimize the loss function:

$$L\left(\theta_q\right) = \left(r_t + \gamma \max_a Q\left(s_{t+1},a;\theta_q\right) - Q\left(s_t,a;\theta_q\right)\right)^2.$$

Similar to classical Q-learning, the agent collects experiences by interacting with the environment. The network trainer constructs a data set $\mathcal{D}$ by collecting the experiences until time t in the form of $(s_{t-1},a_{t-1},r_t,s_t)$. We optimize the loss function $L\left(\theta_q\right)$ using the collected data set $\mathcal{D}$.

In the early stages of training, the agent estimation is not accurate, and a dynamic ϵ-greedy policy is adopted to control the actions, where the agent with a certain probability explores different actions regardless of their reward. This strategy promotes accurate estimation over time, and reduces the risk of overfitting the model to actions with high rewards in the first phase of training.

By substituting the DQN cost function into equation (13), we obtain the FDQN cost as:

$$\min_{\theta_q} L\left(\theta_q\right) = \sum_{i=1}^{N} w_i L_i\left(\theta_{q_i}\right),$$

The overall learning procedure for FDQN is summarized in Algorithm 1.

Algorithm 1 FDQN

Input: Aggregation period Ag, learning rate lr, number of training episodes N_e, episode horizon time T, exploration parameter ϵ, Initial θ_q

Initialization get initial θ_q from server
for $e := 1$ to N_e **do**
  get initial state S
  for $t := 1$ to T **do**
    draw a random number $r \in [0,1]$

$$a_t = \begin{cases} \text{argmax}_a\, Q(s_t,a,\theta_q^e) & \text{if } r > \epsilon \\ \text{pick uniformly action } a & \text{else} \end{cases}$$

    Take action a_t, go to state s_{t+1} and get reward r_{t+1}
    Store$\{a_t,s_t,r_{t+1},s_{t+1}\}$
  end for
  update $\theta_q^{e+1} = \theta_q^e - lr\nabla_{\theta_q} L(\theta_q^e)$
  if $e \mod Ag = 0$ **then**
    send θ_q^t to server for aggregation
    get aggregated θ_q^e from server
  end if
end for

Output: θ_q^*

Fig. 2. Algorithm of the Federated Deep Q Network

VIII. Federated Deep Policy Gradient

In contrast to value based methods such as Q-learning, policy gradient algorithms directly optimize the policy without estimating the Q-values. This approach is more robust to the overestimation problem that affects value based methods. Using a DNN as a function approximator, we define the parametrized policy $\pi\left(a \mid s;\theta_p\right)$, where θ_p are the DNN weights. Herein, we focus specifically on Reinforce as a policy gradient based learning algorithm that updates the policy based on the Monte-Carlo estimation of the agent average return. The objective of Reinforce is defined as:

$$J\left(\theta_p\right) = E_\pi\left\{\pi\left(a \mid s;\theta_p\right) R\right\}$$

given the parameters θ_p and state s, the policy network generates a stochastic policy, that is, a probability vector over actions. Taking the gradient with respect to θ_p we obtain:

$$\nabla_{\theta_p} J\left(\theta_p\right) = E_{\pi_{\theta_p}}\left\{\nabla_{\theta_p} \log\left(\pi\left(a \mid s,\theta_p\right)\right) R\right\},$$

Parameters are updated to increase the probability of actions associated with higher rewards trajectories. Since Reinforce outputs a stochastic policy with non zeros probability over all actions, it does not require an exploration procedure such as the ϵ-greedy strategy described earlier. At the test time, the optimal policy can be obtained by deterministically selecting the action with the largest probability.

By substituting the Reinforce cost function into equation (13) we obtain the FDPG cost:

$$\min_{\theta_p} J\left(\theta_p\right) = \sum_{i=1}^{N} w_i J_i\left(\theta_{p_i}\right)$$

The overall training procedure for FDPG is summarized in Algorithm 2.

Algorithm 2 FDPG

Input: Aggregation period Ag, learning rate lr, number of training episodes N_e, episode horizon time T, Initial θ_p

Initialization get initial θ_p from server
for $e := 1$ to Ne **do**
  get initial state S
  for $t := 1$ to T **do**
    sample action a_t from distribution $\pi(a_t|s_t, \theta_p^e)$
    Take action a_t, go to state s_{t+1} and get reward r_{t+1}
  end for
  update $\theta_p^{e+1} = \theta_p^e + lr\nabla_{\theta_p} J(\theta_p^e)$
  if $e \mod Ag = 0$ **then**
    send θ_p^e to server for aggregation
    get aggregated θ_p^e from server
  end if
end for

Output: θ_p^*

Fig. 3. Algorithm of the Federated Deep Q Network

IX. RESULTS

In this section, we provide a thorough performance evaluation of the proposed federated algorithms. We implement all the models in PyTorch and use the Adam optimizer to train them, setting the learning rate to $lr = 0.001$. We use the same network architecture for DQL and deep policy gradient (Reinforce). We employ a neural network whose number of inputs are equal to the state dimension. The network has two hidden layers with 128 and 64 neurons respectively, followed by Relu activation functions. The output dimension is set to be the same as the number of discretized power levels ($M = 10$ in the results) for policy gradient and DQN. The models will be released in our GitHub repository [1]. In the training procedure, we consider a multi cell network with $N = 25$ cells in which in each cell serves $K = 4$ users. The Doppler frequency and time slot period are set to $f_d = 10$ Hz and $T_s = 20$ ms respectively. We also consider the maximum transmission power to be $P_{\max} = 38\text{dbm}$ and the control parameter in reward function $\beta = 1$.

Figs 2 and 3 depict the average per user rate in the network for centralized, distributed and federated implementations of DPG and DQN algorithms with different aggregation periods.

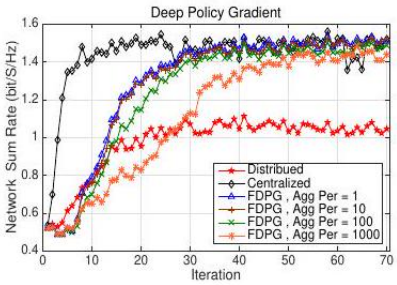

Fig. 4. Comparison of the convergence of distributed deep policy gradients with its federated implementation for different aggregation frequencies

Fig. 2. Comparison of the convergence of distributed deep policy gradients with its federated implementation for different aggregation frequencies.

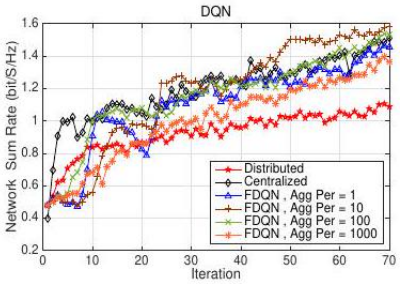

Fig. 5. Comparison of the convergence of distributed deep Q-Learning with its federated implementation for different aggregation frequencies

Fig. 3. Comparison of the convergence of distributed deep Q-Learning with its federated implementation for different aggregation frequencies.

We train each model over 7000 episodes, where each episode itself contains a horizon of $T = 10$ time slots. To plot smoother curves, the results of groups of 100 iterations is reduced to its average in the plots. In the distributed implementation, it is assumed each BS only trains its own model based on the its observed states and does not share the model weights with any other BSs (Agents) or the central server while in the centralized case in each episode all BSs send their user states to the central server in order to learn a global model.

Comparing the results of distributed and federated algorithms, we can see that the achieved rate per user improves significantly in the latter case. Additionally, when comparing the centralized case with federated version we can observe that they have similar performance, but the federated approach the communication between the BSs and server is up to 0.001 times smaller (in $AggPer = 1000$ case). Furthermore, we note that a higher Aggregation frequency leads to faster convergence speed for a penalty of having larger communica-

Fig. 4. Performance Comparison between the federated and distributed deep policy gradient for different number of available cells

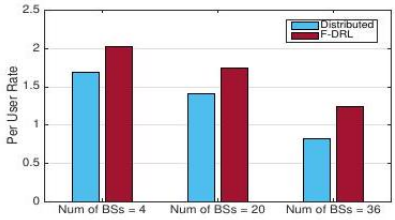

Fig. 6. Performance Comparison between the federated and distributed deep policy gradient for different number of available cells.

tion overhead. It can be observed that FDPG convergence is smoother compared to FDQN for all the different aggregation frequencies, thus providing more coherent performance to users. This can be explained by the use of the ϵ-greedy approach for exploration, which increases the random behavior until convergence is reached, while in the policy gradientbased algorithms, the gradient update selects state-action trajectories associated with higher average rewards which could benefits from model aggregation.

In Fig. 4, the performance of the distributed and federated deep policy gradient algorithms are compared in terms of average per user rate. We can see that as the size of the network increases, the overall interference increases and consequently the average data rate per user decreases. Notably, when the number of BSs is very large (e.g., 36) the gain of federated frameworks over fully distributed optimization is about 40%, while for smaller networks the gain is around 15%.

X. CONCLUSIONS

In this paper, we proposed federated deep reinforcement learning as a tool to solve a distributed power control problem in a wireless multi-cell network. We investigated the performance of DRL with value based and policy based methods and compared their federated version and distributed version implementations. We demonstrated by simulation that aggregating the models of the BSs can improve the performance in terms of overall network sum rate compared to fully distributed implementation, while also being more bandwidth automated networks.

efficient comparing to fully centralized scenarios. We also showed that the F-DRL approach outperforms the conventional optimization-based baselines both in terms of performance and execution time.

XI. REFERENCES

[1] P. Tehrani, F. Lahouti, and M. Zorzi, "Resource allocation in ofdma networks with half-duplex and imperfect full-duplex users," in 2016 IEEE international conference on communications (ICC). IEEE, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," nature, vol. 518, no. 7540 , pp. $529 - 533, 2015$

[3] A. A. Khan and R. Adve, "Centralized & distributed deep reinforcement learning methods for downlink sum-rate optimization," IEEE Transactions on Wireless Communications, 2020 .

[4] F. Meng, P. Chen, L. Wu, and J. Cheng, "Power allocation in multiuser cellular networks: Deep reinforcement learning approaches," $IEEE$ Transactions on Wireless Communications, 2020 .

[5] Y. Sinan Nasir and D. Guo, "Deep actor-critic learning for distributed power control in wireless mobile networks, arXiv e-prints, pp. arXiv$2009, 2020$.

[6] X. Zhang, M. R. Nakhai, G. Zheng, S. Lambotharan, and B. Ottersten, "Calibrated learning for online distributed power allocation in small-cell networks," IEEE Transactions on Communications, 2019 .

[7] 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.

[8] S. Kumar, P. Shah, D. Hakkani-Tur, and L. Heck, "Federated control with hierarchical multi-agent deep reinforcement learning," arXiv preprint arXiv:1712.08266, 2017.

[9] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. Leung, "Attentionweighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," IEEE Journal on Selected Areas in Communications, vol. 39, no. 1, pp. 154-169, 2020.

[10] X. Wang, C. Wang, X. Li, V. C. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," IEEE Internet of Things Journal, 2020 .

[11] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated multi-agent actorcritic learning for age sensitive mobile edge computing," arXiv preprint arXiv:2012.14137, 2020 .

[12] J. Ren, H. Wang, T. Hou, S. Zheng, and C. Tang, "Federated learningbased computation offloading optimization in edge computing-supported internet of things," IEEE Access, vol. 7, pp. 69194-69201, 2019.

[13] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning for industrial internet of things in future industries," arXiv preprint arXiv:2105.14659, 2021.

[14] Y. Zhang, C. Kang, T. Ma, Y. Teng, and D. Guo, "Power allocation in multi-cell networks using deep reinforcement learning," in 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall). IEEE, 2018

[15] F. Meng, P. Chen, and L. Wu, "Power allocation in multi-user cellular networks with deep q learning approach," in IEEE International Conference on Communications (ICC). IEEE, $2019, \text{pp.}1 - 6$.

[16] 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, 2019

[17] H. Ding, F. Zhao, J. Tian, D. Li, and H. Zhang, "A deep reinforcement learning for user association and power

control in heterogeneous networks"" Ad Hoc Networks, vol. 102, p. 102069,2020 .

[18] S. Saeidian, S. Tayamon, and E. Ghadimi, "Downlink power control in dense 5 g radio access networks through deep reinforcement learning," in IEEE International Conference on Communications (ICC), 2020 .

[19] W. Lei, Y. Ye, and M. Xiao, "Deep reinforcement learning based spectrum allocation in integrated access and backhaul networks," IEEE Transactions on Cognitive Communications and Networking, 2020 .

[20] J. Tan, Y.-C. Liang, L. Zhang, and G. Feng, "Deep reinforcement learning for joint channel selection and power control in $\mathrm{d2}$ d networks," IEEE Transactions on Wireless Communications, 2020 .

[21] 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. $4331-4340, 2011$