A Deep Neural Architecture for Real-time Node Scheduling in Uplink Cell-Free Massive MIMO

M. Guenach*+, A. Gorji* and A. Bourdoux*
*Imec Leuven,+TELIN Ghent University
Email: guenach@ieee.org

***Abstract*—This paper proposes a novel real-time and low-complexity solution for the joint power allocation and access point scheduling problem in the uplink cell-free massive multiple-input multiple-output (MIMO) with a serial bandwidth-limited fronthaul architecture. We devise a hybrid optimization framework based on a novel deep neural network architecture for access point scheduling and a low-complexity convex formulation for power allocation. The simulation results demonstrate the effectiveness of the proposed solution in which the trained network exhibits competitive performance compared to state-of-art optimization algorithms, however with a significant computation time reduction. It will be also discussed that the neural-based architecture is very advantageous for dynamic massive MIMO with time-varying number of users.**

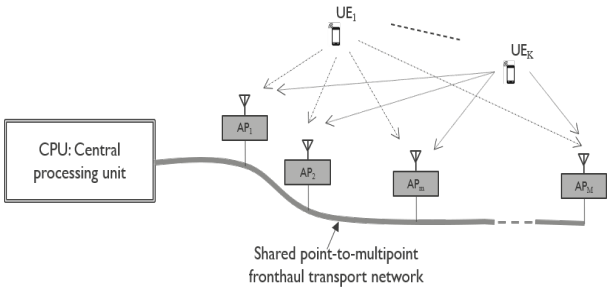

Figure 1. CF m-MIMO with a serial wired fronthaul.

I. Introduction

Cell-free (CF) massive multiple-input multiple-output (m-MIMO) [1][2] is a key technology enabler for 6G and beyond wireless networks featuring low-complexity high-throughput, ultra-reliable and low-latency applications [3]. The user-centric approach enabled by the CF m-MIMO [2] is hindered by the large amount of traffic to be fronthauled between the remote access points (AP)s and the central processing unit (CPU). With the limited bandwidth available in the serial fronthaul (FH) architecture [2][5] with multiple access points (AP)s sharing the same fronthaul wires as depicted in Fig. 1, joint power control and AP scheduling becomes crucial to minimize the co-channel interference and improve the overall spectral efficiency (SE).

Power control using optimal and sub-optimal solutions has been extensively studied for a general CF m-MIMO (see for instance [1][2]). Given the intrinsic complexity involved in most of the optimization based techniques for the power and AP scheduling, the research has evolved towards leveraging machine learning (ML) to replace the complex optimization stage with a well-trained deep neural network. In [7], a multi-layer perceptron (MLP) model trained by an unsupervised ML has been applied to the uplink (UL) CF m-MIMO power control. The proposed algorithm can closely approximate the optimum solutions produced by convex solvers while vastly reducing the complexity. In [8], a (supervised) deep artificial neural network is developed for the UL power control of a CF m-MIMO system. The unsupervised ML scheme in [7] was recently extended to the power allocation problem for the downlink in [9].

Unlike the UL power-only control problem that is convex and can be efficiently solved using popular linear programming solvers [1], the real-time aspect is much more critical in the joint power control and AP scheduling given the non-convexity of the original problem and the non-polynomial complexity with respect to the number of user equipments (UE)s and APs. While efficient architectures have been devised in the literature for the power allocation, no work has been dedicated to the AP scheduling using deep neural networks. In [6], to address the UL CF m-MIMO scalability with a limited bandwidth serial fronthauling architecture, a low-complexity iterative power allocation and AP scheduling algorithm has been proposed that is essentially driven by the channel hardening. Although the proposed algorithm requires a limited number of super-iterations between the power optimization and AP scheduling before the convergence, the time-complexity of the proposed framework is still challenging for real-time deployments of large scale systems.

In this paper, we propose a complexity-efficient deep neural network architecture that undertakes the joint power control and AP scheduling in a near real-time fashion for UL CF m-MIMO. The main contributions of this paper are as follows:

- Design and formulation of the AP scheduling as an image segmentation problem and hiring the well-known U-Net neural network [11] as the solution.
- Proposing a novel objective and training procedure that ensures the neural model produces near-binary decision scores for UE-to-AP assignments, a problem that has not been tackled in the regular U-Net.
- Offering an end-to-end real-time low-complexity joint AP scheduling and power control and demonstrating the efficiency of the architecture in scenarios with time-varying number of UEs.
- A thorough performance and computation-time analysis

of the proposed architecture against a state-of-art convex optimization algorithm and a widely-used nonlinear mixed integer programming solver.

In section II, we briefly introduce the system model. In section III, the optimization algorithm [6] used to feed the artificial neural network (ANN) is summarized. Next, in section IV we cast the AP scheduling as an image segmentation problem and propose an ANN based on the U-Net [11] for this classification problem. Numerical results are provided in section V and conclusions are drawn in section VI.

II. System model

We consider a CF m-MIMO as depicted in Fig. 1 with K UEs and M ($> K$) APs with N antennas per AP. The different APs are interconnected to the CPU through a limited bandwidth serial FH link. We assume a time division duplexing (TDD) protocol of the UL and downlink in which the TDD frame of length τ_C samples is composed of UL training (τ_P samples) and payload (τ_D samples) segments followed by a guard time, and a segment for downlink data.

We assume the same channel estimation protocol as in [1] to acquire the channel $[\mathbf{g}_{m,k}]_n$ between the kth UE and the nth antenna of the mth AP (the subcarrier index is omitted for the sake of notational simplicity). It can be shown that the variance of the channel estimates $\mathrm{E}\{\|\hat{\mathbf{g}}_{m,k}\|^2\}$ is equal to $\alpha_{m,k} = \tau_P\rho\beta_{m,k}^2/(\tau_P\rho\beta_{m,k} + \sigma_w^2)$ where $\beta_{m,k}$ corresponds to the large scale fading, ρ is the maximum transmit power per UE, and σ_w^2 is the additive white Gaussian noise (AWGN) variance. With each AP applying conjugate beamforming to equalize the received noisy data, at most K equalized samples per AP can be reported and the CPU has the task to combine them. With the serial FH architecture in Fig. 1, the normalized bandwidth of the FH wire denoted by $\hat{\mathscr{C}}$ is limited and is, typically, smaller than MK. Hence an optimization algorithm is needed to determine which equalized samples should be reported by each AP. For this purpose, we define the $M \times K$ association matrix $\mathbf{C}$ with the (m,k)th entry $c_{m,k} = 1$ when the CPU receives the kth UE equalized data from the mth AP, and $c_{m,k} = 0$ otherwise. As a matter of fact, it can be shown that the signal to noise ratio (SNR) [6] can be derived as

$$\gamma_k = \frac{N\left(\sum_{m=1}^{M} c_{m,k}\alpha_{m,k}\right)^2 \eta_k}{\sum_{k'=1}^{K}\left(\sum_{m=1}^{M} c_{m,k}\alpha_{m,k}\beta_{m,k'}\right)\eta_{k'} + \frac{\sigma_v^2}{\rho}\sum_{m=1}^{M} c_{m,k}\alpha_{m,k}}. \quad (1)$$

III. Joint access points scheduling and power allocation

We propose, in this paper, to maximize the worst case SNR subject to a per-UE maximum power ρ and to a maximum FH bandwidth $\hat{\mathscr{C}}$ as in [6]:

$$\begin{aligned} &\max_{\boldsymbol{\eta},\mathbf{C}} \min_k \gamma_k(\eta, \mathbf{C}) \\ &\text{st. } 0 \le \eta_k \le 1,\ \forall k \\ &\sum_{m=1}^{M}\sum_{k=1}^{K} c_{m,k} \le \hat{\mathscr{C}} \\ &c_{m,k} \in \{0,1\},\ \forall m,k. \end{aligned} \quad (2)$$

Given i) the existence of binary entries of the association matrix $\mathbf{C}$ and ii) the optimization variables $\{c_{m,k}\}$ and $\{\eta_k\}$ appearing as products in both the numerator and denominator of the SNR given by (1), the above form can be presented as a mixed integer programming problem and, hence, admits a non-convex form [10].

Algorithm 1 Iterative power allocation and AP scheduling.

Init: $\kappa = 0$, set $\mathbf{C} = \mathbf{C}^{(0)}$ and $\boldsymbol{\eta}^{(0)} = optimize(\mathbf{C}^{(0)})$.

1) AP scheduling: $\mathbf{C}^{(\kappa+1)} = optimize(\boldsymbol{\eta}^{(\kappa)})$.
2) Power allocation: $\boldsymbol{\eta}^{(\kappa+1)} = optimize(\mathbf{C}^{(\kappa+1)})$.
3) Increase the super-iteration index: $\kappa \leftarrow \kappa + 1$.
4) Repeat steps **(1)**-**(3)** until convergence.

The formed optimization problem in (2) was tackled in [6] by formulating a sequence of linear programming (LP) problems. The master algorithm, as depicted in Alg. 1, then iterates between two steps, i) finding the optimal power coefficients assuming a known association matrix and ii) AP scheduling given the known power coefficients from the previous step. While the power allocation can be cast as a linear-program, in [6], the AP scheduling has been also formulated as an LP by relaxing the binary constraints in (2) and proposing an iterative channel hardening balancing (IHB) procedure (see Alg. 3 in [6]). Results in [6] verify the capability of the new IHB algorithm in solving (2) and providing competitive performance compared to off-the-shelf combinatorial optimization solvers.

While the proposed algorithm in [6] shows a descent performance improvement in contrast to the state-of-the-art, the iterative nature of the AP scheduling part may make the overall architecture not immediately applicable for real-time deployments with time varying system configurations.

IV. ANN based solution for access-point scheduling problem

In this paper, we propose a novel architecture sketched in Fig. 2 that is based on ANN to solve the AP scheduling problem knowing the input system parameters, while the power allocation problem can be still solved using either LP-based optimization techniques [1] or near-optimal solutions [4], which both provide near real-time solutions for the power allocation problem.

A. Problem statement and model architecture

Let $\mathbf{X}$ denote a matrix of size $M \times K$ that is formed by taking the set of parameters $\boldsymbol{\beta}$ as the input to the network.

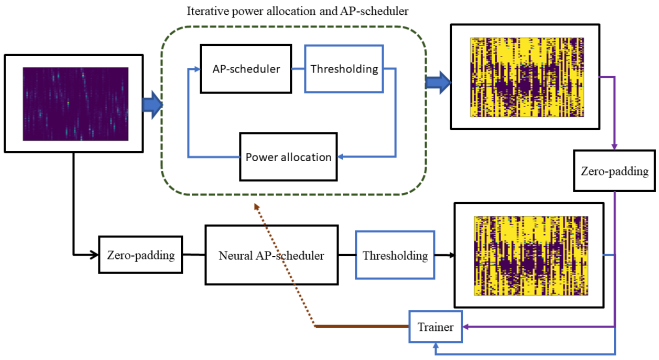

Figure 2. The overall optimization architecture.

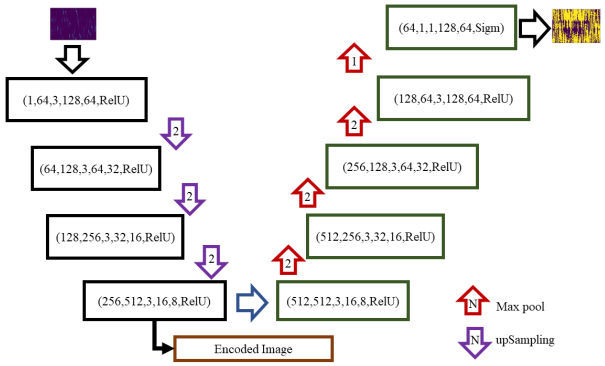

Figure 3. The proposed architecture for AP prediction using an encoder/decoder like network.

The architecture employed in this paper as depicted in Fig. 3 is inspired by a classification task handled by the U-Net network first proposed in [11]. The network receives a single-channel $\boldsymbol{\beta}$ input passing through a number of convolutional operations followed by the non-linearity that leads to the contraction of the input image and the creation of an encoded image. Each rectangular block or hidden layer in the ANN of Fig. 3 is represented by a tuple (a, b, c, d, e, f) that demonstrates a convolutional layer where a and b are the number of input channels and output filters, respectively, c denotes the kernel size, the tuple (d, e) corresponds to the size of the input image, and f is the type of the non-linearity. The encoded image is then sent to the expansion phase that aims to reconstruct an image in the same size of the input matrix.

The network output will be now represented by a probability matrix $\mathbf{Y}$ of size $M \times K$ where the (m, k)th entry of the matrix is denoted by a *sigmoid* function that measures the probability of assigning the mth AP to the kth UE. Assuming $\mathbf{Y}$ as the $M \times K$ output matrix, the corresponding (m, k)th entry can be written as follows:

$$\boldsymbol{Y}_{mk} = p(c_{mk} = 1|\mathbf{X}; \theta) = \frac{1}{1 + \exp(-\boldsymbol{a_{mk}}[\boldsymbol{\theta}])}, \quad (3)$$

where $\boldsymbol{a_{mk}}[\boldsymbol{\theta}]$ denotes the (m, k)th activation function that is constructed by applying the input matrix $\mathbf{X}$ to both contraction and expansive layers with $\boldsymbol{\theta}$ being the parameters of the network.

B. Training Process

Having a set of N_P training tuples $(\mathbf{X}^n, \bar{\mathbf{C}}^n)$ where $\bar{\mathbf{C}}^n$ corresponds to the true association matrix, the objective is to maximize the cross-entropy loss [12] for the proposed ANN architecture of Fig. 3 as follows:

$$E_{\boldsymbol{\theta}} = \sum_n \sum_{m,k} \log \left[\begin{array}{l} \bar{c}^n_{mk} p(c_{mk} = 1; \boldsymbol{\theta}) + \\ (1 - \bar{c}^n_{mk}) \times (1 - p(c_{mk} = 1; \boldsymbol{\theta})) \end{array} \right] \quad (4)$$

with $\bar{c}^n_{mk}$ being the true label ($\bar{c}^n_{mk} \in \{0, 1\}$) associated with the (m, k)th pixel in the nth training example, and $p(c_{mk} = 1; \boldsymbol{\theta})$ being the same probability defined by (3) where the input $\boldsymbol{X}$ has been omitted for the sake of notational brevity. We also propose to address the binary constraints in the original problem (2) by adding the regularizer to the objective of neural training as:

$$\tilde{E}_{\boldsymbol{\theta}} = E_{\boldsymbol{\theta}} + \lambda \sum_n \min(2 f^n_{mk}(\boldsymbol{\theta}), -2 f^n_{mk}(\boldsymbol{\theta}) + 2) \quad (5)$$

where $f^n_{mk} = \bar{c}^n_{mk} p(c_{mk} = 1; \boldsymbol{\theta}) + (1 - \bar{c}^n_{mk}) \cdot (1 - p(c_{mk} = 1; \boldsymbol{\theta}))$ and λ denotes a hyper-parameter that tunes the weight of the binary constraints in the training objective.

Parameters of the model can be now learned using the stochastic gradient descent (SGD) or the SGD with a momentum such as ADA-Grad [12]. A summary of the training procedure can be also found in Alg. 2 with the algorithm delivering the association matrix, allocated power and the corresponding SNR.

C. Threshold selection

The trained model with parameters $\boldsymbol{\theta}$ is used to produce probability scores for each entry of the association matrix $p(c_{mk} = 1; \theta)$. The association matrix $\mathbf{C}$ is now formed by applying a threshold over each column of the output probability matrix as follows: $c^{\tau_k}_{mk} = 1$ if $p(c_{mk} = 1; \boldsymbol{\theta}) \geq \tau_k$, and $c^{\tau_k}_{mk} = 0$ otherwise with τ_k being the kth user-specific threshold. The thresholds can be obtained by solving the following constrained optimization problem:

$$\begin{aligned} \max_{\tau_k} & \sum_m \left[\begin{array}{l} c^{\tau_k}_{mk} p(c_{mk} = 1; \boldsymbol{\theta}) \\ +(1 - c^{\tau_k}_{mk}) \times (1 - p(c_{mk} = 1; \boldsymbol{\theta})) \end{array} \right] \\ \text{st.} & \sum_m c^{\tau_k}_{mk} \leq \hat{M}_k, \end{aligned} \quad (6)$$

with $\sum_k \hat{M}_k = \hat{\mathscr{C}}$.

Algorithm 2 Training algorithm for the ANN considering the max/min budget constraints

Init: Initial ANN parameters $\boldsymbol{\theta}^{(0)}$ and hyper-parameter λ
For epochs e from 0 to e_{max}

1) For batch b from 0 to b_{max}
   a) Generate a set of L tuples $\{(\mathbf{X}^l, \bar{\mathbf{C}}^l), l \in 0, \ldots, L\}$ by a random selection of input $\mathbf{X}^l$ and running the iterative optimization [6] to obtain the association $\bar{\mathbf{C}}^l$
   b) Update the ANN weights using the objective in (5) and the ADAM algorithm [12]
2) Stop if either the maximum number of epochs is reached or the loss change over subsequent batches is trivial

Algorithm 3 Training data generation for a given budget and a range of served users

Init: Number of samples n_K per system load K, maximum budget $\hat{\mathscr{C}}$, and training instances $t = \{\}$
For K from 4 to 64
For n from 0 to n_K

1) Generate $\boldsymbol{\beta}_K^{(n)}$ and compute the corresponding $\boldsymbol{\alpha}_K^{(n)}$.
2) Run the iterative optimization algorithm from [6] to obtain the power $\boldsymbol{\eta}_K^{(n)}$ and association matrix $\bar{\mathbf{C}}_K^{(n)}$.
3) Append the triple $\{\boldsymbol{\beta}_K^{(n)}, \boldsymbol{\eta}_K^{(n)}, \bar{\mathbf{C}}_K^{(n)}\}$ to the training set t

V. Numerical Results

We consider the same piazza topology as in [6] wherein the APs are placed along the perimeter of a $[100 \times 100]$ m^2. The large scale fading coefficient $\beta_{m,k} = PL_{m,k} \cdot SF_{m,k}$ depends on the path-loss $PL_{m,k}$ and the shadow fading $SF_{m,k}$ and are modeled in the same way as in [1][6]. We choose $M = 128$, a varying number of UEs K and a maximum FH budget $\hat{\mathscr{C}} = 60\%$ $(= 0.6 \cdot M \cdot K)$.

A. Training and Performance Evaluation

We train the neural-network with the ANN architecture and using the training sequences provided by the optimization algorithm proposed in [6]. The input to the ANN is a 128×64 image and any training instance whose number of UEs is less than 64 is zero-padded to become compatible with the original input. The training data are generated according to the procedure given by Alg. 3. For this round of experiments, n_K $(= 1000$ for $K \leq 16$ and 500 otherwise) instances are generated per system load Kthat give 40.000 training tuples in total. For the hyper-parameter optimization, we also choose values of $\lambda \in \{0, 1, 3, 10, 100\}$.

Performance evaluation is done using a set of new instances produced by Alg. 3. Having the association matrices by both the neural-model and the ground-truth (iso near-optimal solution from [6]), as a performance metric, the resulting worst SNR is calculated for all the test cases and the cumulative distribution function (cdf) of the log-scaled error between the neural-model and the ground-truth, i.e. $\Delta_\gamma = \gamma_{OPT} - \gamma_{ML}$, is depicted in Fig. 4 for different values of K where $\lambda = 3$ is chosen as the value of the regularizer that provides the best results.

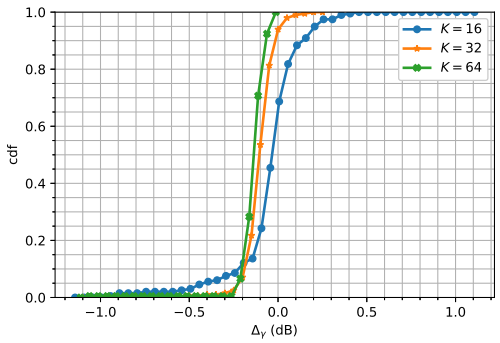

Figure 4. The cdf of error in SNR for the optimization algorithm and ML solution $\Delta_\gamma = \gamma_{OPT}$-$\gamma_{ML}$.

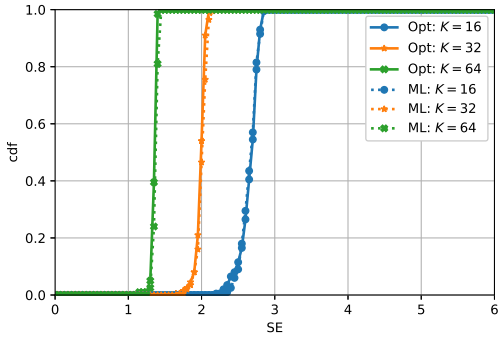

Figure 5. Data rate cdf from both the ML and the optimization algorithm versus K.

Results in Fig. 4 show that the neural-model produces association matrices leading to SNRs that in 95% of test cases lie in a 0.2 dB error interval. The results also verify that the performance of the ANN model degrades when the number of users falls down. As Fig. 4 demonstrates, the percentage of test cases falling in the 0.1 dB interval falls from 100 in $K = 64$ to 90 when $K = 16$ users is chosen. Therefore, the neural-model shows near-perfect alignment with the ground-truth when the number of users grows, which is in-line with the purpose of AP scheduling for the m-MIMO system with a large number of users. The corresponding SE results shown in Fig. 5 also reveal a good match with the SNR error from Fig. 4. The SEs produced by both the ANN and the optimization are almost identical demonstrating again the the good match between the two approaches.

B. Time Complexity and Time-varying User Case

To study the computational efficiency of the proposed architecture, and as the benchmark, an off-the-shelf near globally optimal NMIP [13] solver is used to address the joint power allocation and AP scheduling problem. A new

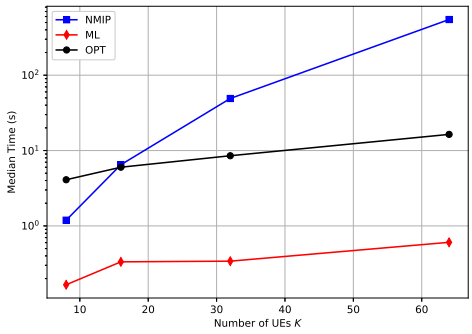

Figure 6. Computational time versus K for the ML model, the optimization algorithm of [6] and the NMIP solver.

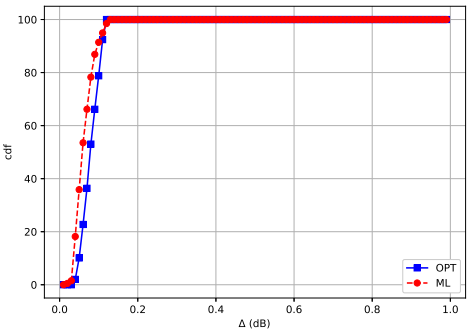

Figure 7. The cdf of error in SNR $\Delta_\gamma = \gamma_{NMIP}$-$\gamma_{\{ML/OPT\}}$ when compared against the NMIP solver as the benchmark.

set of test instances is now created for different values of $K \in \{2^l\}, l \in \{4, 5, 6\}$. For each instance, both the computational time and the resulting SNRs are calculated for three algorithms namely 1) the optimization in [6], 2) the NMIP solver, and 3) the proposed neural-model. To calculate the timing, all the processing are conducted on the same Intel Core i5-8350U CPU with the clock-frequency of 1.7 GHz.

The computational time medians from the three schemes versus the number of UEs shown in Fig. 6 reveal a near-polynomial complexity of the NMIP solver versus the number of users while the optimization algorithm has been proven to exhibit linear complexity as justified in [6]. The novel neural-model shows near-constant complexity with the computational time being ~ 100 times lower than that of the optimization algorithm for $K = 64$. While the discrepancy seems less critical for $K \leq 16$, it is evident from Fig. 6 that the NMIP algorithm becomes infeasible for a large-system when going beyond 32 UEs. Although the optimization algorithm provides ~ 10 times better efficiency to the NMIP, it still shows some linear complexity that will make it less-competitive for real-time applications with a large number of time-varying users.

In addition, all the UE-specific SNRs are collapsed to form a single vector of SNRs and, assuming the NMIP as the benchmark, the histogram of the error is derived and its corresponding cdf is now shown in Fig. 7. On top of the much superior time efficiency, the neural-model is able to deliver competitive performance to the near-optimal NMIP where 95% of test-instances report errors that fall in a 0.1 dB region. Indeed, the results in this section clearly show that the novel ANN-based architecture can output near-optimal solutions that are very close to the standard solvers in terms of accuracy while the superb computational efficiency of the model makes it a very attractive choice for real-time applications with a time-varying number of UEs.

VI. Conclusions

This paper proposed a novel architecture based on ANNs and linear programming to address the challenging AP scheduling and power allocation problems respectively in the UL of CF m-MIMO systems. A novel neural-model was developed to generate the association matrix with the model being trained by the data generated using a near-optimal optimization framework. Simulation results show that beside the competitive performance to the ground-truth and an off-the-shelf NMIP solver as the benchmark, the ANN enjoys much less computational complexity that makes it a perfect choice for real-time scenarios with a time-varying system load. The proposed architecture is agnostic to the radio topology and therefore is applicable to any CF m-MIMO system.

References

[1] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta, "Cell-Free Massive MIMO versus Small Cells," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1834–1850, March 2017.

[2] G. Interdonato, E. Björnson, H. Q. Ngo, P. Frenger and E. G. Larsson, "Ubiquitous Cell-Free Massive MIMO Communications," *EURASIP Journal on Wireless Communications and Networking*, August 2019.

[3] P. Popovski, C. Stefanovic, J. J. Nielsen, E. D. Carvalho, M. Angjelichinoski, K. F. Trillingsgaard, and A. S. Bana, "Wireless Access in Ultra-Reliable Low-Latency Communication (URLLC)," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5783-5801, August 2019.

[4] E. Nayebi, A. Ashikhmin, T. L. Marzetta,H. Yang, and B. D. Rao, "Precoding and Power Optimization in Cell-Free Massive MIMO systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 7, pp. 4445-4459, July 2017.

[5] P. Zhang, F. M. J. Willems, "On the Downlink Capacity of Cell-Free Massive MIMO with Constrained Fronthaul Capacity," Entropy 2020, 22, 418.

[6] M. Guenach, A. Gorji and A. Bourdoux, "Joint Power Control and Access Point Scheduling in Fronthaul-constrained Uplink Cell-Free Massive MIMO Systems," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2709-2722, April 2021.

[7] R. Nikbakht, A. Jonsson and A. Lozano, "Unsupervised-Learning Power Control for Cell-Free Wireless Systems," *IEEE PIMRC*, September 2019.

[8] C. D. Andrea, A. Zappone, S. Buzzi and M. Debbah, "Uplink Power Control in Cell-Free Massive MIMO via Deep Learning," *IEEE CAMSAP*, December 2019.

[9] R. Nikbakht, A. Jonsson and A. Lozano, "Unsupervised-Learning Power Allocation for the Cell-Free Downlink," *IEEE ICC*, June 2020.

[10] S. Boyd, and L. Vandenberghe, "Convex Optimization," *Cambridge University Press*, March 2004.

[11] O. Ronneberger, F. Fischer, and T. Brox, "U-Net: convolutional networks for biomedical image segmentation," *Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, 2015.

[12] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," *MIT Press*, 2016.

[13] Open-source Manual, "Introduction to Ipopt: A tutorial for downloading, installing, and using Ipopt," April 2015.