MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM) | 979-8-3503-7423-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/MILCOM61039.2024.10773886

Semi-decentralized Message Allocation for Cell-Free Networks using Unsupervised Learning

Noel Teku Ravi Tandon
Department of Electrical and Computer Engineering
University of Arizona, Tucson, Arizona 85719
E-mail: {*nteku1*, *tandonr*}@arizona.edu

Tamal Bose
EpiSci
Poway, California, 92064
E-mail: tbose@episci.com

***Abstract*—In this paper, we present an unsupervised learning framework for message allocation in Cell-free networks (CFNs) for latency minimization. One of the key features of CFNs is that users' data can be decoded by multiple access points (APs), i.e., in a "cell-free" manner by letting users connect to multiple APs simultaneously; leading to the problem of message allocation across APs. While a fully centralized approach for message allocation can make the most out of the flexibility offered via CFNs, it requires prohibitive coordination overhead. In this paper, we instead propose a novel semi-decentralized machine learning based framework for message allocation. It allows each user to split their messages using a "learned" model (e.g., a neural network) which takes two inputs: a user's local channel gains and aggregate global SINRs at the APs.**

The model is trained with the objective of minimizing the latency of the network. To accomplish this, the total latency derived from the model's learned message split for each user is the main component of the loss used to update the model. Different methods are investigated for training the model by presenting variations of how the latency is computed. We use the cumulative distribution function (CDF) of latency as the key performance metric and compare our proposed semi-decentralized approach against several centralized methods as well as uniform and greedy message allocation techniques. Our results indicate that the semi-decentralized machine learning based method can approach the performance of the centralized methods with very little coordination overhead and outperforms greedy/uniform allocation methods.

I. Introduction

Cell-Free networks (CFNs) have been proposed to deal with the challenges of dense cellular networks, including higher interference and complexity [1]. They become critical as wireless standards have encouraged incorporating frequency reuse strategies in cellular networks to conserve spectrum; however, if the same frequencies were shared across cells, it could result in increased intercell interference [2]. This is especially worse for cell-edge user equipment (UEs) as they would experience increased intercell interference [3]. A concrete example of a CFN in the literature is Cell-Free MIMO, where instead of organizing access points (APs) and UEs into cells, the APs are spread over an area, while connected to a CPU, and are responsible for serving every UE in that area [4] [5]. Cell-Free MIMO can also be considered as a more efficient form of Coordinated Multipoint with Joint Transmission (CoMP-JT) due to reduced overhead requirements enabled via time division duplexing [6].

Several architectures have been proposed for implementing CFNs. In [7], varying levels of interaction between the APs and the CPU are investigated for cell-free MIMO, ranging from fully centralized to fully distributed. Generally, centralized operation refers to the CPU managing the system using the signals received at the APs; contrarily, distributed operation involves the APs serving the UEs [7]. In [7], for example, centralized operation is implemented by having the CPU handle channel estimation and data decoding; however, distributed operation is implemented as a small-cell network where the APs use their local channel estimates for decoding, and decoding duties for a UE are determined based on which AP can provide it the highest spectral efficiency. There has also been significant work in user-centric Cell-Free MIMO, where a UE is served only by a certain number of APs based on a particular feature [8].

A. Main Contributions

Given the flexibility of CFNs, the design of message allocation schemes is critical to optimize overall network performance and spectral efficiency. In this work, a message allocation framework is presented where each AP is designated a certain amount of data from each UE to decode; these message allocations are determined with the objective of minimizing the total latency of the CFN. To accomplish this, we propose a novel *semi-decentralized machine learning based approach*. While a fully centralized approach for message allocation can make the most out of the flexibility offered via CFNs, it requires prohibitive coordination overhead (in terms of global channel state information (CSI)). Instead, in our proposed approach, the machine learning (ML) model operates locally at every user and takes two inputs: a) Coarse global information (aggregate signal-to-interference-plus noise ratios (SINRs) from APs from previous blocks) and b) current local CSI from the user.

Another key challenge we address is the training procedure for such semi-decentralized networks. Supervised learning cannot be directly applied in this setting, as it would require a-priori knowledge on the latency incurred for a particular split of a UE's message for multiple channel conditions. Instead,

This work was supported by NSF grants CAREER 1651492, CCF-2100013, CNS-2209951, CNS-1822071, CNS-2317192, and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing under Award Number DE-SC-ERKJ422, and NIH Award R01-CA261457-01A1.

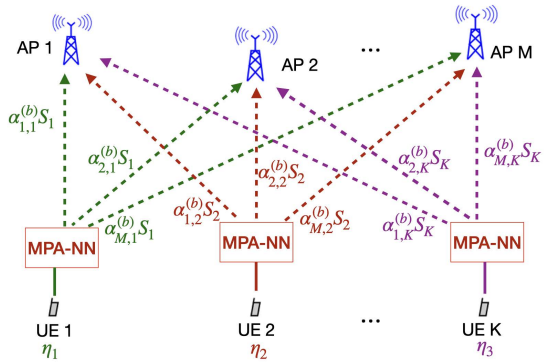


Fig. 1: Illustration of uplink message/power allocation problem in Cell-free networks for K UEs and M APs. Message and power allocation parameters $\{\alpha_{m,k}^{(b)}, \eta_{m,k}^{(b)}\}$ are generated for each UE by a local model (i.e. MPA-NN) to minimize the overall latency.

we train the model via an unsupervised learning approach, where the latency derived from the model's message allocation decision(s) is treated as the main component of the loss used to update the model. The latency is composed of both computational and transmission latencies. We study two variations on computing the transmission latency, namely average and worst case (maximum) latency across the system. We compare our approach against fully centralized methods (Interior Point (IP) and Sequential Quadratic Programming (SQP) optimization algorithms) as well as uniform and greedy message allocation approaches for various path loss models. Our results highlight the effectiveness of the semi-decentralized machine learning based approach and show that it comes quite close to the performance of fully centralized methods with very little coordination overhead and outperforms greedy/uniform methods[1].

B. Related Works

Rate-splitting, where messages are split into public/private portions that are decoded differently, for the downlink of a Cell-Free MIMO system was investigated in [9] [10]. Additionally, works in the literature have proposed cell-free MIMO as a framework for Mobile Edge Computing (MEC), which can be considered a form of message allocation, while addressing latency in varying degrees. In [11], an optimization problem is formulated, assuming a user-centric Cell-Free MIMO setup, for assigning UEs transmit powers and deciding how much of an AP's (i.e. edge server) and CPU's (i.e. central server) resources are assigned for serving each UE. A latency constraint is explicitly considered, incorporating the computational, transmission (between UE and APs), and fronthaul (between APs and CPU) latencies, while assuming that signaling/feedback latencies are implicitly covered under this formulation. In [12], a user-centric cell-free MIMO MEC system is analyzed via a metric called the successful computation probability (SCP), representing how likely the computational latency of a task is smaller than a certain deadline. In [13], two optimization problems for latency and energy consumption formulations are derived for a user-centric Cell-Free MIMO MEC network, where the latency formulation explicitly includes the downlink latency for retrieving the processed bits from the cloud.

In [14]— [16], approaches were presented for a MEC user-centric Cell-Free MIMO setup using deep reinforcement learning (DRL). In [14], a decentralized approach was proposed, where the DRL technique was implemented at each user. In [15] [16], this assumption is expanded on by having the DRL techniques at each UE share information with each other during training but not during testing, making it a hybrid centralized/decentralized approach. Latency is included as part of the input to the DRL technique (as a single deadline, whose success in being met is represented by a binary value [14] and as user-specific deadlines [15] [16]); however, the primary objective of these works are to minimize energy consumption.

II. System Model

A. System Operation

We consider a CFN with M APs, $m = 1, 2, ..., M$ and K UEs for $k = 1, 2, ..., K$ as shown in Figure 1. We consider uplink transmission from the UEs to the APs, where each AP is responsible for decoding a certain fraction of data from each UE. It is assumed that the UEs conduct slotted transmissions to the APs and that the CFN is synchronized. We assume that time is organized in blocks, where a block encapsulates the slotted transmissions from the UEs to the APs and the decoding of the data at the APs. This is shown in Figure 2(a). Additionally, the channels between the UEs and the APs are assumed to be constant over a block and can vary across blocks. At each block, the k^{th} UE wants to send a message of size S_k bits. The set of fractions the k^{th} UE uses to assign portions of the S_k bits to each AP is denoted as $\boldsymbol{\alpha}_k^{(b)} = [\alpha_{1,k}^{(b)}, \alpha_{2,k}^{(b)}, ..., \alpha_{M,k}^{(b)}]$, where b represents the block index, $0 \leq \alpha_{j,k}^{(b)} \leq 1$, and $\sum_j \alpha_{j,k}^{(b)} = 1$. For the k^{th} UE, we denote the codeword generated for each sub-message as $x_{j,k}^{(b)}$, where $x_{j,k}^{(b)}$ encodes the $\alpha_{j,k}^{(b)}$ fraction of the k^{th} UE's message. The power available at the k^{th} UE (P_k) is allocated to each codeword for transmission using power control factors denoted as $\boldsymbol{\eta}_k^{(b)} = [\eta_{1,k}^{(b)}, \eta_{2,k}^{(b)}, ..., \eta_{M,k}^{(b)}]$, where $0 \leq \eta_{m,k}^{(b)} \leq 1$ and $\sum_j \eta_{j,k}^{(b)} = 1$. Each UE then transmits $X_k = \sqrt{P_k} \sum_{j=1}^{M} \sqrt{\eta_{j,k}^{(b)}} x_{j,k}$. The portion of the k^{th} UE's signal that the m^{th} AP is responsible for decoding is: $\sqrt{P_k \eta_{m,k}^{(b)}} g_{m,k}^{(b)} x_{m,k}^{(b)}$, where $g_{m,k}^{(b)}$ represents the channel between the m^{th} AP and the k^{th} UE.

There are two sources of interference the m^{th} AP experiences when decoding the k^{th} UE's signal. The first source is from the k^{th} UE sending codewords intended for the other $(M-1)$ APs, which is given as: $\sum_{j \neq m,}^{M} \sqrt{P_k \eta_{j,k}^{(b)}} g_{j,k}^{(b)} x_{j,k}^{(b)}$. The second is from the other $(K-1)$ UEs sending the portions of their respective messages across all M APs, which is given as: $\sum_{j=1}^{M} \sum_{k' \neq k,}^{K} \sqrt{P_{k'} \eta_{j,k'}^{(b)}} g_{j,k'}^{(b)} x_{j,k'}^{(b)}$. The signal received by

[1]The simulation code for this paper is available on https://github.com/nteku1/Cell_Free_Code

the m^{th} AP in the b^{th} block can then be written as follows:

$$y_m^{(b)} = \underbrace{\sqrt{P_k \eta_{m,k}^{(b)}} g_{m,k}^{(b)} x_{m,k}^{(b)}}_{\text{Desired Signal of } k^{th} \text{ UE}} + \underbrace{\sum_{j \neq m,}^{M} \sqrt{P_k \eta_{j,k}^{(b)}} g_{j,k}^{(b)} x_{j,k}^{(b)}}_{\text{Self-interference of } k^{th} \text{ UE}} + \underbrace{\sum_{j=1}^{M} \sum_{k' \neq k,}^{K} \sqrt{P_{k'} \eta_{j,k'}^{(b)}} g_{j,k'}^{(b)} x_{j,k'}^{(b)}}_{\text{Interference from } (K-1) \text{ UEs}} + \omega_m, \quad (1)$$

where ω_m represents additive noise. During the first block, because the optimal $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ are not known a-priori, we randomly initialize the $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ for each UE (while also normalizing them by their respective sums). The $\boldsymbol{\eta}_k^{(b)}$ are then used to calculate the uplink SINR between the k^{th} UE and m^{th} AP as follows:

$$\text{SINR}_{m,k}^{(b)} = \frac{P_k \eta_{m,k}^{(b)} |g_{m,k}^{(b)}|^2}{1 + \sum_{j=1}^{M} \sum_{k' \neq k}^{K} P_{k'} \eta_{j,k'}^{(b)} |g_{j,k'}^{(b)}|^2}. \quad (2)$$

Because of this constraint, the SINRs passed into the model are always derived from the previous block, as shown in Figure 2(a). However, because it is assumed that channel estimation is conducted at the beginning of subsequent blocks, the UEs have access to local CSI (i.e. knowledge of the channel gains between itself and the M APs) during the current block. The set of channel gains between the k^{th} UE and the M APs is denoted as $\boldsymbol{g}_k^{(b)} = [|g_{1,k}^{(b)}|, |g_{2,k}^{(b)}|, ..., |g_{M,k}^{(b)}|]$. The set of coarse aggregate SINRs at the APs from the previous block is denoted as $\mathbf{SINR}^{(b-1)} = [\text{SINR}_1^{(b-1)}, \text{SINR}_2^{(b-1)}, ..., \text{SINR}_M^{(b-1)}]$. These two vectors, as shown in Figure 2(b), constitute the input to the *message and power allocation model*. The UEs then split their messages using their $\boldsymbol{\alpha}_k^{(b)}$ obtained from the model and transmit these portions using their $\boldsymbol{\eta}_k^{(b)}$ to the respective APs for processing.

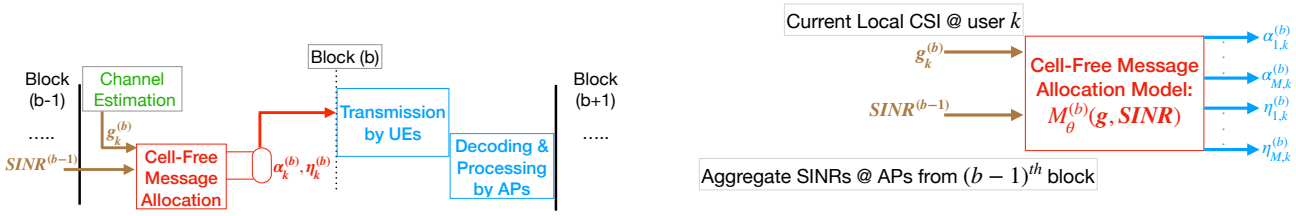
B. Latency Formulations

We adopt the same method for computing the computational and transmission latencies as in prior works [14] [17]. The largest transmission latency from the k^{th} UE to the M APs is as follows: $L_k^{\text{Tran}} = \max_m(\frac{\alpha_{m,k}^{(b)} S_k}{B \log_2(1+\text{SINR}_m)})$ where B denotes the channel bandwidth. The average and maximum/worse-case transmission latencies of the CFN are denoted by $\bar{L}_{Tran} = \frac{\sum_{k=1}^{K} L_k^{\text{Tran}}}{K}$ and $L_{Tran}^{\text{Max}} = \max_k(L_k^{\text{Tran}})$ respectively. The computational latency of the M^{th} AP is as follows: $L_m^{\text{Comp}} = \frac{(\sum_{k=1}^{K} \alpha_{m,k}^{(b)} S_k) c_m}{f_m}$ where $\sum_{k=1}^{K} \alpha_{m,k}^{(b)} S_k$ represents the portions of data from the K UEs that the m^{th} AP is responsible for decoding, c_m is the number of cycles for the m^{th} AP to process one sample, and f_m is the processing frequency of the m^{th} AP. Similar to [14], the computational latency term indicates how the m^{th} AP's services are split across the UEs. The formulations of the total latency for the CFN used in this work are as follows: $L_1^{\text{Max}} = \bar{L}_{Tran} + L_{Comp}^{\text{Max}}$ and $L_2^{\text{Max}} = L_{Tran}^{\text{Max}} + L_{Comp}^{\text{Max}}$, where $L_{Comp}^{\text{Max}} = \max_m(L_m^{\text{Comp}})$ represents the largest computational latency across the M APs. The model is evaluated by adopting L_1^{Max} and L_2^{Max} to analyze how well it can perform under different latency scenarios. The models are assumed to be placed near/called by the UEs so that their respective $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ can be acquired quickly. In doing so, latencies associated with control signaling (e.g. passing information to the model and delivering the $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ to their respective UEs) are assumed to be negligible. *The objective of this effort is to learn a message and power allocation model with the goal of minimizing L_1^{Max} and L_2^{Max}.*

III. Semi-Decentralized Machine Learning based Message Allocation

Motivation and overview: As stated in Section I, the proposed method of latency minimization in this work is a semi-decentralized machine learning approach. The objective is to enable UEs to make decisions on how their messages should be split across APs that minimize the total latency of the CFN. This approach is presented as a balance between two extreme modes of operation. On one extreme, the model could exclusively use coarse global information (i.e. accessible to all UEs) to perform message allocation but this would come at the cost of significant overhead. On the other extreme, the model could exclusively use local CSI but, in doing so, would suffer significant performance degradation. In this work, to strike a balance between the two extremes, each UE uses the same model, which is given $\mathbf{SINR}^{(b-1)}$ (i.e. coarse global information) and $\boldsymbol{g}_k^{(b)}$ (i.e. fine local information) as input as shown in Figure 2(b). The output of the model, after post-processing (described in more detail later), is then used as the $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ for the k^{th} UE. The $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ for each of the K UEs are generated at every block with the objective of minimizing either $L_1^{\text{Max}}(M_\theta^{(b)})$ or $L_2^{\text{Max}}(M_\theta^{(b)})$, where M_θ and θ represents the model and model parameters respectively.

Training Methodology: First, the model parameters, message allocation $\boldsymbol{\alpha}_k^{(0)}$, and power allocation $\boldsymbol{\eta}_k^{(0)}$ for the K UEs are initialized. Channel gains are sampled from the initial block and are used with $\boldsymbol{\eta}_k^{(0)}$ to calculate the initial SINRs at the M APs. During the next block, the current channel gains and the SINRs from the previous block (i.e. $\mathbf{SINR}^{(0)}$) are passed into the model to determine the message and power allocation, $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ for each UE. The learned $\boldsymbol{\eta}_k^{(b)}$ and channel gains are used to re-calculate the SINRs, which are then used with the learned $\boldsymbol{\alpha}_k^{(b)}$ to calculate $L_1^{\text{Max}}(M_\theta^{(b)})$ or $L_2^{\text{Max}}(M_\theta^{(b)})$. The loss function is as follows (assuming $L_1^{\text{Max}}(M_\theta^{(b)})$): $\boldsymbol{L}^{\mathbf{grad}} = \nabla L_1^{\text{Max}}(M_\theta^{(b)})$. $\boldsymbol{L}^{\mathbf{grad}}$ is the gradient of the subsequent total latency derived from the message/power allocations for each UE. This implies that the model is applied in an unsupervised learning context as no a-priori labels are used to guide its training process. $\boldsymbol{L}^{\mathbf{grad}}$ is then passed to an Adam optimizer to update the model's weights, and the procedure continues for T blocks. For each UE, only the top τ $\alpha_{m,k}^{(b)}$ are used for message allocation, where τ is a hyperparamater restricting how many APs a UE can connect to. τ is used to limit the overhead a UE needs for message allocation. The indices of the



(a) System Operation In Blocks.

(b) Cell-Free Message Allocation Model Workflow.

Fig. 2: (a) In one block, SINR from previous block and CSI from current block are passed to the model to determine message allocation. UEs split/transmit messages to appropriate APs for decoding using $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ determined by the model. (b) Cell-Free Message Allocation model using coarse aggregate SINRs at the APs from the previous block and local CSI from the current block at the UEs to learn $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$.

top τ $\alpha_{m,k}^{(b)}$ are used to select which $\eta_{m,k}^{(b)}$ are used for that UE. The resulting $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ are normalized by their respective sums. The training procedure is summarized in Algorithm 1.

Algorithm 1 Training for Semi-Decentralized Message Allocation Model

Initialize $\theta^{(0)}$ (Model Parameters)
Initialize ϵ (learning rate), ρ_1 & ρ_2 (exponential decay rates), δ (small constant), $\boldsymbol{s} = \boldsymbol{r} = 0$
Initialize $[\boldsymbol{\alpha}_1{}^{(0)}, ..., \boldsymbol{\alpha}_K^{(0)}]$ (Initial Message Allocation)
Initialize $[\boldsymbol{\eta}_1^{(0)}, ..., \boldsymbol{\eta}_k^{(0)}]$ (Initial Power Allocation)
Calculate $\mathbf{SINR}^{(0)}$ at APs using gains from channel model and initial power allocation: $[\boldsymbol{g}_1^{(0)},..., \boldsymbol{g}_K^{(0)}, \boldsymbol{\eta}_1^{(0)},..., \boldsymbol{\eta}_k^{(0)}]$
for $b = 1, ...T$ **do**
  Sample $[\boldsymbol{g}_1^{(b)}, \boldsymbol{g}_2^{(b)}, ..., \boldsymbol{g}_K^{(b)}]$ from channel model
  $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)}) = M_\theta^{(b-1)}(\boldsymbol{g}_k^{(b)}, \mathbf{SINR}^{(b-1)})$
  Get τ largest elements from $\boldsymbol{\alpha}_k^{(b)}$ and normalize.
  Get corresponding indices from $\boldsymbol{\eta}_k^{(b)}$ and normalize.
  Compute SINRs using $[\boldsymbol{g}_1^{(b)},..., \boldsymbol{g}_K^{(b)}, \boldsymbol{\eta}_1^{(b)},..., \boldsymbol{\eta}_k^{(b)}]$
  Compute Latency using Message Allocation decided by model: $L_1^{\text{Max}}(M_\theta^{(b-1)})$ (assuming L_1^{Max})
  Update $\theta^{(b)}$ using Adam on loss function [20]:
  $\boldsymbol{s} = \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{L}^{\mathbf{grad}}$
  $\boldsymbol{r} = \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{L}^{\mathbf{grad}} \odot \boldsymbol{L}^{\mathbf{grad}}$
  $\theta^{(b)} = \theta^{(b-1)} - \epsilon \frac{\frac{\boldsymbol{s}}{1-\rho_1^t}}{\delta + \frac{\boldsymbol{r}}{1-\rho_2^t}} \boldsymbol{L}^{\mathbf{grad}}$
end for

IV. Simulation Results and Discussion

A. Datasets & Simulation Setup

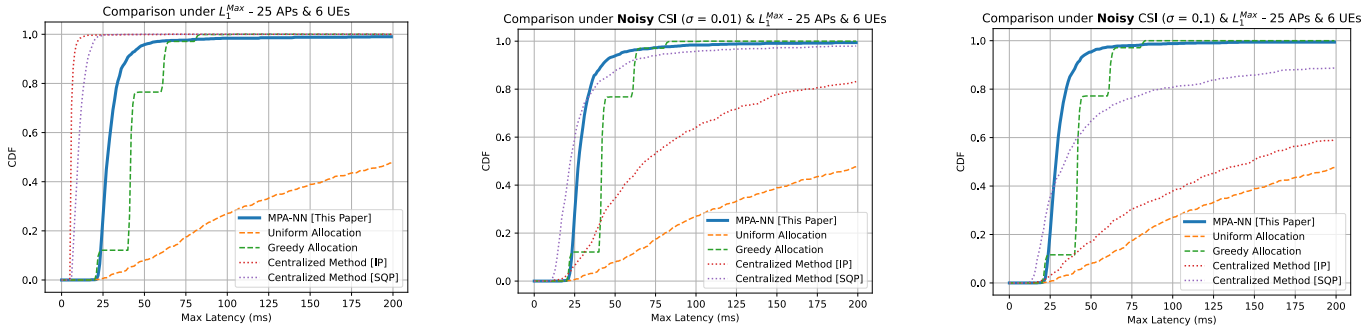
For simulation and validation, CFNs with two different path loss scenarios are considered. The three-slope path loss $(PL_{m,k})$ model used in [5] is as follows:

$$PL_{m,k} = \begin{cases} -L\text{-}15\log_{10} d_1\text{-}20\log_{10} d_0 & d_{m,k} \le d_0 \\ -L\text{-}15\log_{10} d_1\text{-}20\log_{10} d_{m,k} & d_0 < d_{m,k} \le d_1 \\ -L\text{-}35\log_{10} d_{m,k} & d_{m,k} > d_1, \end{cases}$$

where $d_{m,k}$ is the distance between the m^{th} AP and the k^{th} UE, d_0 and d_1 are 10 and 50 m respectively, and L is a constant that depends on the assumed carrier frequency and height of the antennas of the APs and UEs [5]. If the path loss $-L - 35\log_{10}(d_{m,k})$ is chosen, shadow fading is added [7]. The path losses are applied to a spatially correlated Rayleigh fading channel which results in $g_{m,k}^{(b)}$ [7] as used in (1) and (2). The path loss is also normalized by the noise power, set to $-92dBm$ as in [7]. We perform experiments on two scenarios, each assuming a 100m x100m grid. The first scenario uses the full three-slope path model. The second scenario only uses the path loss $-L-35\log_{10}(d_{m,k})$. To keep the spirit of the model, shadow fading was added for any $d_{m,k}$ greater than 50 m.

We adopt a similar experimental setup as the prior works: [5] [7] [17] [18] [19]. We use a neural network as our message and power allocation model (abbreviated as MPA-NN) to show the results of the proposed approach using Tensorflow/Keras. The MPA-NN's input layer is $2M$ neurons, taking in M SINRs and M channel gains between a UE and the APs. The output layer is $2M$ neurons to assign an $\alpha_{m,k}^{(b)}$ and $\eta_{m,k}^{(b)}$ for each AP, with a sigmoid activation. Unless otherwise stated, $\tau = 2$. The MPA-NN has 2 hidden layers and its learning rates are chosen from $[0.002, 0.0002, 0.00002]$. Other attributes (hidden activations, dropout, $\ell 1/\ell 2$-norm regularization) are varied for each CFN. For all results, the methods were tested for 1000 blocks (with the result of the first block excluded as it is used to get the initial SINRs). In all CFNs, $f_m = 1MHz$, $c_m = 20\frac{cycles}{sample}$, $S_k = 1000$, $P_k = 100mW$, $B = 20MHz$, and the carrier frequency is 1.9 GHz. Each AP and UE has a single antenna and their positions are fixed. More simulation details are presented in the code.

We compare the MPA-NN with four approaches. The first is uniform allocation; equal amounts of data are sent to each AP with equal power (e.g. for 10 APs, $\alpha_{m,k}^{(b)} = \eta_{m,k}^{(b)} = 0.1$ $\forall m, k$). The second approach is greedy allocation, where the AP with the largest channel gain ($|g_{m,k}^{(b)}|$) with respect to a UE is assigned to decode that UE's signal, with full power used in that transmission. The third and fourth approaches are the Interior Point (IP) and Sequential Quadratic Programming (SQP) methods, which act as our centralized approaches, using Matlab's in-built optimization toolbox. The centralized methods have access to the global CSI of the K UEs and can calculate the SINRs using (2); their objective is to determine the $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ for the K UEs that minimize the total latency. The CDF of the total latency is the metric for all simulations.



(a) (b) (c)

Fig. 3: Comparison of all approaches under L_1^{Max} (latency) formulation for noisless CSI (a) and noisy CSI (b), (c) with CSI noise standard deviations of $\sigma = 0.01$ and $\sigma = 0.1$ respectively. MPA-NN is more robust to noisy CSI than centralized methods.

B. Impact of CSI Error

In this section, we study the impact of imperfect CSI on our proposed method by corrupting the local CSI available at the UEs with complex Gaussian noise, distorting the SINRs and local channel gains passed into the model. Figure 3(a) shows the performance of the MPA-NN, centralized (IP, SQP), uniform, and greedy methods assuming the single path loss and L_1^{Max} formulation for a CFN of 25 APs and 6 UEs with noiseless CSI. It indicates that the MPA-NN is more likely to ensure lower latencies than uniform and greedy allocation. The centralized methods outperform all methods but require more overhead. Figures 3(b) & 3(c) show the performance of the techniques assuming noisy CSI for the same CFN with noise standard deviations of $\sigma = 0.01$ and $\sigma = 0.1$ respectively. For the MPA-NN, the same neural network structure is used but now trained on noisy CSI instead of noiseless CSI. In both setups, the centralized methods suffer performance loss compared to when noiseless CSI is assumed. The MPA-NN, however, in both setups, maintains a very similar performance to when noiseless CSI is assumed. The MPA-NN outperforms and performs comparably with the centralized IP and SQP methods respectively in ensuring relatively low latencies with high probability. Greedy allocation is also robust in both scenarios but is still suboptimal compared to the MPA-NN.

C. Comparison with Other CFN Message Allocation Methods

In this section, we study the impact of using the different latency formulations L_1^{Max} and L_2^{Max} on our proposed model. Figures 4(a) & 4(b) show results for a CFN of 10 APs and 6 UEs under the three-slope path loss model, with $\tau = 3$. Figure 4(a) shows results under the L_1^{Max} formulation. It indicates that with a smaller number of APs, each method performs reasonably well. The MPA-NN can still ensure lower latencies with a higher probability compared to uniform and greedy allocation. Figure 4(b) shows results under the L_2^{Max} formulation. All methods still perform reasonably well except uniform allocation, which suffers a significant loss in performance compared to the L_1^{Max} formulation. This hints at the L_2^{Max} formulation being more stringent compared to L_1^{Max}. The MPA-NN still ensures lower latencies with a higher probability compared to uniform and greedy allocation. In both setups, the centralized methods outperform all other methods but require significantly more overhead. Figure 4(c) shows the performance of the techniques for a CFN of 20 APs and 6 UEs. The MPA-NN attains a high probability of ensuring lower latencies compared to uniform and greedy allocation as it starts to outperform both at approximately 13 ms. The MPA-NN also outperforms the IP method after approximately 29 ms and performs closely to the SQP method with less overhead.

D. Complexity & Impact of Scaling up CFN network size

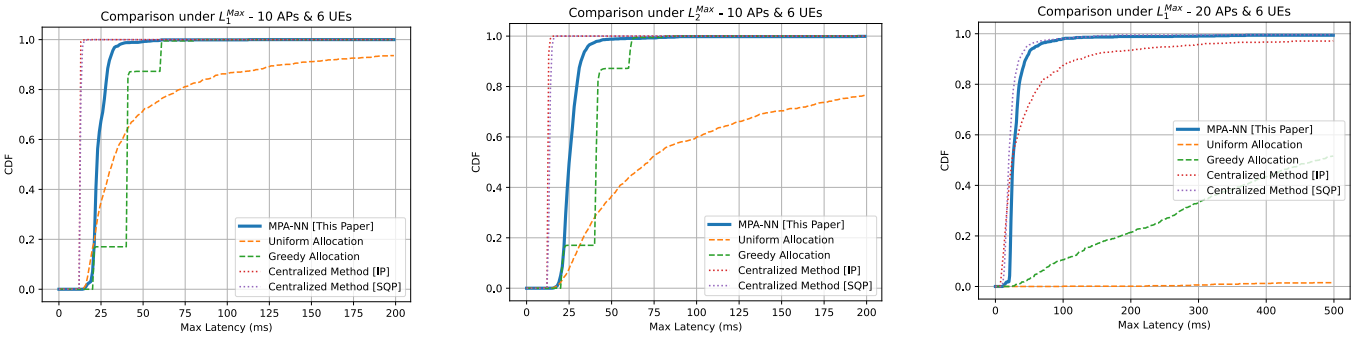
Figure 5(a) shows the results of a CFN with 50 APs and 6 UEs under the single path loss scenario to represent a scaled up CFN ($M >> K$). It indicates that even with more APs to split the message, the greedy and uniform based methods perform relatively worse to the MPA-NN. While the centralized SQP method outperforms all methods, the MPA-NN performs comparably with the centralized IP method at approximately 40 ms. We also compare the time complexity of the MPA-NN with the centralized methods. Table I shows the time required for the methods to generate $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ for the test sets of CFNs with 10, 20 and 50 APs. The values reported were averaged over 5 runs of each test set. The table indicates that the MPA-NN is significantly faster in generating $(\boldsymbol{\alpha}_k^{(b)}, \boldsymbol{\eta}_k^{(b)})$ compared to the centralized methods, showing that it is more efficient for message allocation. Figure 5(b) provides the means and standard deviations (in ms) on the latencies incurred on the test sets of each CFN with the L_1^{Max} formulation, further validating that the MPA-NN can approach and outperform the performance of the centralized methods when noisless and noisy CSI are assumed respectively.

Method/APs	**MPA-NN [Ours]**	IP	SQP
10 APs	$\mathbf{6.73 \pm 0.054}$	90.91 ± 0.3	121.66 ± 0.29
20 APs	$\mathbf{5.97 \pm 0.083}$	112.43 ± 0.22	298.52 ± 1.26
50 APs	$\mathbf{7.46 \pm 0.089}$	256.76 ± 0.56	1600.44 ± 14.15

TABLE I: Average inference times (in seconds) over 5 runs of proposed and centralized methods for CFNs with 10, 20 & 50 APs.

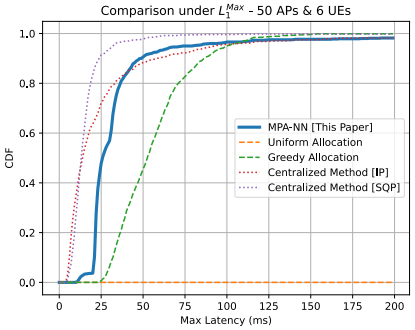
V. Conclusion

In this paper, we presented a semi-decentralized learning approach for message and power allocation for CFNs. An approach for training this model was detailed with the incurred latency being the main component of the loss function. Results were presented evaluating the approach on different path loss models. Our approach almost always outperforms uniform



(a) (b) (c)

Fig. 4: Comparison of all approaches under two different latency optimization objectives for the three-slope path loss scenario. Figure (a) shows the comparison for L_1^{Max} and (b) for L_2^{Max}. In (c), we consider single path loss scenario under L_1^{Max}. MPA-NN significantly outperforms uniform and greedy allocations and approaches the performance of centralized (IP, SQP) methods.



(a)

Avg. Latency ± SD. (in ms)		MPA-NN (This Paper)	Centralized		Decentralized	
			IP	SQP	Uniform	Greedy
Noiseless CSI	10 APs	24.26 ± 5.64	**12.2 ± 0.2**	12.7 ± 0.604	107.26 ± 703.57	39.73 ± 11.15
	20 APs	40.14 ±114.04	199.7 ± 1199.5	**28.1 ± 67.9**	144685.37± 630978.73	717.13 ± 747.51
	25 APs	34.85 ± 57.63	**6.1 ± 2.5**	10.8 ± 3.8	1248.29 ± 5755.76	44.54 ± 13.15
	50 APs	50.68 ± 276.66	83.1 ± 884.7	**16.1 ± 21.2**	29728.61 ± 272765.82	56.16 ± 22.85
Noisy CSI (25 APs)	$\sigma = 0.01$	**33.4 ± 30.58**	289.9 ± 1611.5	60.1 ± 584.3	1248.29 ± 5755.76	44.48 ± 13.12
	$\sigma = 0.1$	**33.69 ± 41.36**	1097.9 ± 5956.3	201.3 ± 1223.5	1248.29 ± 5755.76	44.5 ± 12.9

(b)

Fig. 5: (a) Impact of scaling up CFN on various approaches as we increase the number of APs to 50. (b) Means and standard deviations of latencies incurred for test sets of CFNs under various scenarios including both noiseless and noisy CSI.

and greedy-based methods and performs close to centralized methods. Furthermore, the approach remains robust even when the size of CFN scales and is also robust to noisy CSI.

REFERENCES

[1] Y. Al-Eryani, M. Akrout and E. Hossain, "Multiple Access in Cell-Free Networks: Outage Performance, Dynamic Clustering, and Deep Reinforcement Learning-Based Design," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 4, pp. 1028-1042, April 2021.

[2] Y. Zhou et. al, "An overview on intercell interference management in mobile cellular networks: From 2G to 5G," *IEEE International Conference on Communication Systems*, pp. 217-221, 2014.

[3] K. Yang, "Interference management in LTE wireless networks [Industry Perspectives]," *IEEE Wireless Communications Journal*, vol. 19, no. 3, pp. 8-9, June 2012.

[4] J. Zhang et. al, "Cell-Free Massive MIMO: A New Next-Generation Paradigm," *IEEE Access*, vol. 7, pp. 99878-99888, 2019.

[5] 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.

[6] G. Interdonato, E. Björnson, H. Q. Ngo, et al., "Ubiquitous cell-free Massive MIMO communications". *EURASIP Journal on Wireless Communications and Networking*, Article number: 197, 2019.

[7] E. Björnson and L. Sanguinetti, "Making Cell-Free Massive MIMO Competitive With MMSE Processing and Centralized Implementation," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 77-90, Jan. 2020.

[8] H. A. Ammar et. al, "User-Centric Cell-Free Massive MIMO Networks: A Survey of Opportunities, Challenges and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 611-652, 2022.

[9] A. R. Flores, R. C. De Lamare and K. V. Mishra, "Rate-Splitting Meets Cell-Free MIMO Communications," *IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 657-662, 2022.

[10] A. Mishra et. al, "Rate-Splitting Assisted Massive Machine-Type Communications in Cell-Free Massive MIMO," in *IEEE Communications Letters*, vol. 26, no. 6, pp. 1358-1362, June 2022.

[11] G. Interdonato and S. Buzzi, "The Promising Marriage of Mobile Edge Computing and Cell-Free Massive MIMO," *IEEE International Conference on Communications*, Seoul, Korea, pp. 243-248, 2022.

[12] S. Mukherjee and J. Lee, "Offloading in Edge Computing-Enabled Cell-Free Massive MIMO Systems," *IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, pp. 1-6, 2018.

[13] G. Femenias and F. Riera-Palou, "Mobile Edge Computing Aided Cell-Free Massive MIMO Networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1246-1261, Feb. 2024.

[14] F. D. Tilahun, A. T. Abebe and C. G. Kang, "Joint Communication and Computing Resource Allocation over Cell-Free Massive MIMO-enabled Mobile Edge Network: A Deep Reinforcement Learning-based Approach," *International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, South Korea, pp. 344-346, 2021.

[15] F. D. Tilahun, A. T. Abebe and C. G. Kang, "Multi-Agent Reinforcement Learning for Distributed Resource Allocation in Cell-Free Massive MIMO-Enabled Mobile Edge Computing Network," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 12, pp. 16454-16468, Dec. 2023.

[16] F. D. Tilahun, A. T. Abebe and C. G. Kang, "DRL-based Distributed Resource Allocation for Edge Computing in Cell-Free Massive MIMO Network," *IEEE Global Communications Conference*, Rio de Janeiro, Brazil, pp. 3845-3850, 2022.

[17] T. T. Vu et. al, "Cell-Free Massive MIMO for Wireless Federated Learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6377-6392, Oct. 2020.

[18] https://keras.io/examples/rl/ddpg_pendulum/

[19] Y. Zhao, I. G. Niemegeers and S. M. H. De Groot, "Dynamic Power Allocation for Cell-Free Massive MIMO: Deep Reinforcement Learning Methods," *IEEE Access*, vol. 9, pp. 102953-102965, 2021.

[20] I. Goodfellow, et al. Deep Learning, *MIT Press*, 2016.