IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 April 2023, accepted 5 May 2023, date of publication 9 May 2023, date of current version 15 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3274556

RESEARCH ARTICLE

Neural Network Aided User Clustering in mmWave-NOMA Systems With User Decoding Capability Constraints

ADITYA S. RAJASEKARAN[1,2], (Member, IEEE), AND HALIM YANIKOMEROGLU[1], (Fellow, IEEE)

[1]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada
[2]Ericsson Canada Inc., Ottawa, ON K2K 2V6, Canada

Corresponding author: Aditya S. Rajasekaran (aditya.sriram.rajasekaran@ericsson.com)

The work of Aditya S. Rajasekaran was supported by Ericsson Canada Inc. in the form of tuition assistance.

ABSTRACT This paper proposes a computationally efficient two-stage machine learning based approach using neural networks to solve the cluster assignment problem in a millimeter wave-non orthogonal multiple access (mmWave-NOMA) system where each user's individual successive interference cancellation (SIC) decoding capabilities are taken into consideration. The artificial neural network (ANN) is applied in real time to assign users to clusters taking each user's instantaneous channel state information (CSI) and SIC decoding capabilities as inputs. The algorithm is trained offline on cloud resources, i.e., not using the base station (BS) compute resources. This training is done using a dataset obtained by offline computation of input parameters using the optimization algorithms called NOMA-minimum exact cover (NOMA-MEC) and NOMA-best beam (NOMA-BB) from our earlier work in this area. As a result, we term the proposed algorithms in this paper as ANN-NOMA-MEC and ANN-NOMA-BB, respectively. The problem with applying optimization techniques, even low-complexity heuristics, in live networks for user clustering is that they require a very large number of computation steps to make a clustering decision. If these clustering decisions are based on the instantaneous channel of hundreds of users, it becomes prohibitively complex to implement in practical systems on a millisecond granularity as required by beyond 5G (B5G) systems. Instead, our proposed approach takes all this complexity offline and even off the BS compute resources and instead only applies a trained neural network to make such clustering decisions at a microsecond granularity on hundreds of users. Simulation results show the effectiveness of the ANN-NOMA-MEC and ANN-NOMA-BB schemes as the neural network trained on offline simulation data performs comparably with the NOMA-MEC and NOMA-BB heuristics that is applying computationally intensive algorithms to make every clustering decision in a live network.

INDEX TERMS Non-orthogonal multiple access (NOMA), successive interference cancellation (SIC), neural networks (NN), millimeter-wave (mmWave), user clustering (UC).

I. INTRODUCTION

Non-orthogonal multiple access (NOMA), when run in the mmWave spectrum has the ability to serve a large number of connected users at a time in a high bandwidth spectrum, two key requirements for beyond fifth-generation (B5G) communication systems. Such systems are referred to as mmWave-NOMA systems. The mmWave channels are highly correlated, allowing for opportunities to form clusters of spatially correlated users to be served by a single beam and separated in the power domain through NOMA [1], [2], [3]. Two key aspects of achieving a good performance in any NOMA system and more so in mmWave-NOMA systems where the spatial correlation amongst users can be exploited for efficient cluster formation to increase system throughput are user clustering and user ordering [4]. User clustering is the selection of users to serve in a NOMA clusters, while user ordering is the order in which the successive

The associate editor coordinating the review of this manuscript and approving it for publication was Fan-Hsun Tseng.

interference cancellation (SIC) procedure is applied amongst these clustered users.

The user clustering and ordering aspects are also important in terms of the amount of processing the end-user is required to undertake in a downlink (DL) NOMA system. Based on the selected user ordering in a NOMA cluster, the strongest user has to decode the signals of all other users in the SIC procedure while the weakest user only has to decode its own signal. It is clear that the number of users in the cluster and the position in which any one user is placed in the SIC decoding order determines how many other users signals a user needs to decode in that cluster. In our prior work in this area in [1], we termed this limitation as the SIC decoding capability of the user and factored it into the user clustering scheme. In a DL NOMA system, this SIC decoding capability of a user equates to the number of other users signals the said user is capable of decoding before decoding its own signal. This SIC decoding capability number of each user is influenced by its hardware and end-user capabilities [5], [6]; an IoT device might not be able to decode any other user's signals in the SIC procedure while high-end smartphones could be capable of processing upwards of ten other user's signals. As motivated in [1], it is important to consider each users SIC decoding capability into the clustering and ordering problem; as if not, we could end up in cluster formations where users need to decode the signals of more users than they have the hardware capability for which leads to infeasible solutions to implement in practice. As a result, in [1], we considered a heterogeneous NOMA system as one where each user has its own SIC decoding capability. A homogeneous system was also discussed in [1] as a system where all users can be considered to have the same SIC decoding capability.

A. RELATED WORK

Optimizing the sum-rate of mmWave-NOMA systems involves considering the user clustering, user ordering, beamforming and power allocation schemes [4]. In [3], these aspects were jointly optimized for a downlink (DL) mmWave-NOMA systems. Another common approach is to divide the user clustering and ordering, beamforming and power allocation schemes into separate sub-problems. Since we want to factor in each users individual SIC decoding capability into the clustering, we focus on this type of approach where the user clustering and ordering aspects are tackled first and then the power allocation is optimized for a given set of cluster formation. For example, the user clustering approach in NOMA systems presented in [15] formulates the Karush–Kuhn–Tucker (KKT) conditions in a rate maximization optimization problem that assumes a fixed size of clusters. In fact, limiting the number of users in a cluster is a typical approach to limit the SIC decoding requirements put on the end user in a homogeneous system, e.g., [7], and in [16], the optimum cluster sizes from a performance perspective is analyzed. Within a cluster, users are typically ordered based on their effective channel gains to maximize throughput and satisfy each users QOS requirement [17]. When it comes to mmWave-NOMA systems, several user clustering schemes have been proposed in the literature that exploit the high correlation amongst user channels in mmWave systems [3], [18] by clustering correlated users together to maximize system throughput [18], [19], [20]. The cosine similarity metric is often used to determine the level of correlation either between users [2], [8], between users and random beams in [18] or between users and fixed beam directions in [1]. Thus, these works in mmWave-NOMA area frame a rate optimization problem the user clustering and ordering aspects as the parameters to be optimized and solve them by low complexity heuristics that typically involve using the cosine similarity metric.

One issue with solving optimization problems, even low-complexity heuristics, in live networks for user clustering is that they require a very large number of computation steps to make a clustering decision. If these clustering decisions are based on the instantaneous channel of hundreds of users, it becomes prohibitively complex to implement in practical systems on a millisecond granularity as required by beyond 5G (B5G) systems. This is where machine learning has been looked into as an enabler to solve the user clustering problem in NOMA systems. The work in [21] classified user clustering problems in mmWave-NOMA systems into joint resource aware user clustering techniques such as the ones applying the cosine similarity metric as described above and a second class of algorithms called learning assisted user clustering techniques to bring down the complexity. In [21], the complexity of several user clustering schemes in the mmWave-NOMA literature is analyzed and it is shown that a significant run-time complexity is inherited by all schemes to make clustering decisions on a millisecond granularity, especially as the network size grows. However, the machine learning techniques applying K-means like clustering algorithms bring down the complexity compared to traditional optimization schemes.

There are two main machine learning based themes of work that are relevant to the discussion in this paper. The first is the theme of work in [2], [8], [9], [10], where the user clustering and ordering problem in mmWave-NOMA systems is solved using an unsupervised clustering ML approach. These works exploit the high correlation amongst users' channels and the fact that mmWave propagation is dominated by the LoS path to effectively employ K-means clustering. In [11], an advancement on k-means for the cluster formation problem is proposed. Since the effects of multipath propagation are limited in mmWave spectrum, the user clustering in mmWave-NOMA systems comes down to finding spatially correlated users with the available CSI at the BS. A concrete clustering metric based on channel correlation among users is proposed in [22]. This is precisely what unsupervised clustering algorithms are capable of achieving without any labeled training data. On a similar line, in [23], a multi-label classification problem is framed to solve the user clustering

TABLE 1. Comparison of this paper with existing literature.

Papers		System Model		Clustering Problem Formulation		
Category	References	Fixed users/cluster	SIC decoding capability	ML with offline training	ML with online clustering	Low complexity heuristics
SIC decoding paper	[1]	✔	✔	✘	✘	✔
Normal clustering papers	[7]	✔	✘	✘	✘	✔
Unsupervised clustering papers	[2], [8]–[11]	✔	✘	✘	✔	✘
Supervised clustering papers	[12]–[14]	✔	✘	✔	✘	✘
This paper		✔	✔	✔	✘	✘

problem. One other theme of work in general NOMA systems is that from Kumaresan et al. in [12], [13], and [14]. The authors in [14] propose a ANN-based user clustering framework that learns from a training data set obtained through simulation settings and a brute force search approach of all possibilities [12]. The dataset is used to train the neural network that is then applied to the clustering decisions in the network. In [13], the same authors extended the machine learning scheme to instead use the extreme machine learning (ELM) method to solve the clustering problem. Finally, user clustering solutions in other NOMA based system models such as one incorporating device-to-device communication proposed in [24] also run into similar problems of exponential run-time complexity to solve the clustering problem in a live network.

One important limitation with these schemes is that there is no flexibility incorporated to account for the SIC decoding capability limitations of each individual user. To address this, in our previous work in this area in [1], we proposed a user clustering and ordering scheme for a mmWave-NOMA system that takes into consideration the SIC decoding capability of each individual user in the system. The work in [1] frames the user clustering and ordering problem as a cluster minimization problem that requires the algorithm to consider each users SIC decoding capability. To this end, two heuristics are proposed in [1]:

1) A NOMA-minimum exact cover (NOMA-MEC) algorithm for heterogeneous systems.
2) A NOMA-best beam (NOMA-BB) algorithm for homogeneous systems.

In this work, we use these algorithms to build up the labelled training dataset for our proposed neural network to learn from, as discussed next.

B. MOTIVATION AND CONTRIBUTIONS

In order to actually deploy mmWave-NOMA systems, practical considerations need to be factored in. One example is the availability of instantaneous CSI of large number of users, and the complexity involved in making clustering decisions with this large amount of CSI information. In order to mitigate the availability of CSI, location aided [25] and vision aided [26] clustering techniques have been proposed in the literature as well as techniques to combat imperfect CSI availability [27]. However, in this work we will assume the full CSI of each user is available to the BS. Even so, a user clustering and ordering problem (UCOP) in mmWave-NOMA systems that relies on the instantaneous channel information of hundreds of users, needs to find the right balance between performance and complexity while considering practical limitations such as the SIC decoding capability of the user and the availability of CSI in order to be usable in practical deployments. In terms of performance, the UCOP scheme needs to generate a clustering result that maximizes or close to maximizes the system throughput while satisfying each users minimum QOS constraint. However, to lead to a feasible solution, the UCOP needs to factor in the SIC decoding capabilities of the users in the system. For a homogeneous system, that amounts to just limiting the number of users per cluster while for heterogeneous systems, it means accounting for each users SIC decoding capabilities. The NOMA-MEC and NOMA-BB schemes proposed in [1] addressed these requirements but the complexity of the algorithms was quite large to be run at a millisecond granularity. As shown in [1], depending on the parameter settings, up to 60,000 computation steps need to be executed to make a clustering decision every millisecond. Due to the latency sensitive nature of this computation, it cannot be offloaded to the cloud or an external entity as the round-trip delay would be too large. With low-cost and small cell BSs being increasingly studied for B5G systems, there is a need to manage the computational resource needs available at the cell site. To this end, we propose a novel machine learning based approach using neural networks to factor in the SIC decoding capabilities of the users and build a low-complexity scheme that produces clustering results on par with NOMA-MEC and NOMA-BB for heterogeneous and homogeneous systems, respectively. We distinguish our work from the other machine learning user clustering schemes in the NOMA literature by factoring in the SIC decoding capabilities of the users into the machine learning scheme. For example, compared to the unsupervised clustering schemes in [2], [8], and [9], we use a supervised learning scheme as it is a better fit to the clustering problem once SIC decoding capabilities are considered. Compared to the supervised learning schemes presented in [12], [13], and [14], we distinguish ourselves again by studying a neural network that is capable of forming clusters while factoring in the SIC decoding capabilities of the users which these other works do not take into consideration. Table 1 illustrates how the system model and clustering problem formulation of this paper is distinguished from existing work in the literature.

The proposed neural network is trained offline on labelled data samples generated using the NOMA-MEC and NOMA-BB algorithms. Concretely, for different possible inputs of user positions, channels and SIC decoding

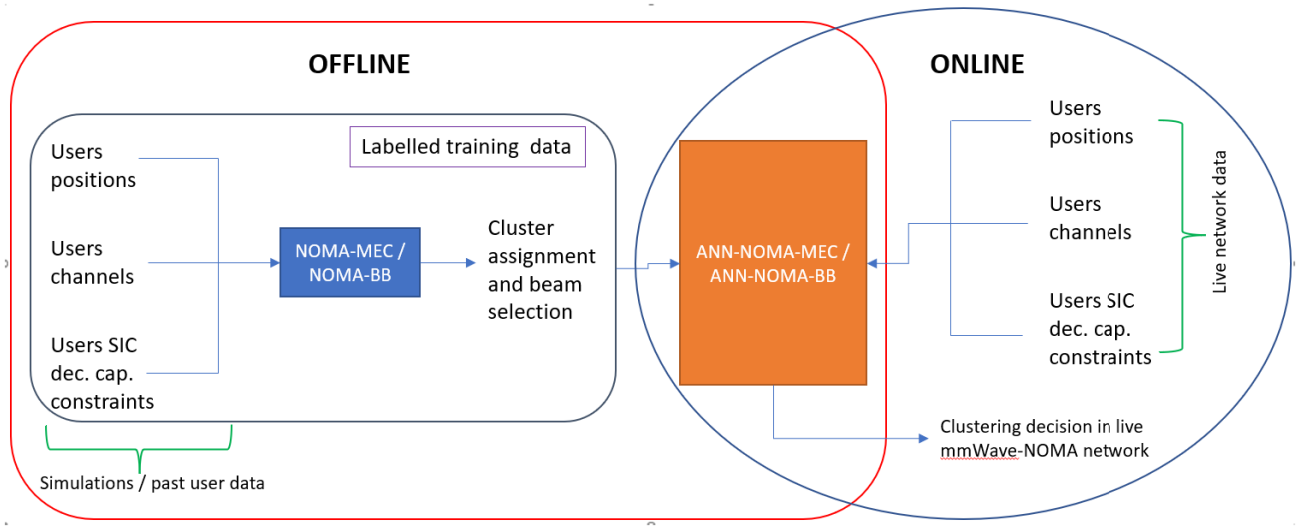


FIGURE 1. Illustrating the proposed ANN-NOMA-MEC and ANN-NOMA-BB networks learning offline from labelled training data obtained using the NOMA-MEC and NOMA-BB algorithms and then applied to real user inputs in a live network.

capabilities, the NOMA-MEC or NOMA-BB algorithm is run and a labelled data set is generated. This labelled dataset is used to train the neural netowrk. As a result, we term the two proposed algorithms in this paper as ANN-NOMA-MEC and ANN-NOMA-BB, for heterogeneous and homogeneous systems, respectively. This training of the neural netowrk can be done offline from the BS, possibly on cloud compute resources. The trained neural network is then applied in the live network with the input comprising of the users channels and SIC decoding capabilities. This two-stage flow is illustrated in Fig. 1.

Applying this trained neural network model directly on the users inputs allows for it to be run at the millisecond granularity that is required to make optimal clustering decisions as users channels change or users enter or leave the system. Users SIC decoding capabilities can also change over time as a function of the battery level of the users. Hence, this model continues to make a clustering decision considering a whole fresh set of input every millisecond or so, but in directly applying a trained model to do so, this is a far more reasonable cost to take at the BS. Our contributions in this paper can thus be summarized as follows:

- We propose two ANNs for user clustering factoring in SIC decoding capability constraints in mmWave-NOMA systems, namely ANN-NOMA-MEC and ANN-NOMA-BB, for heterogeneous and homogeneous systems respectively, that are trained offline using labelled datasets generated from running the NOMA-MEC and NOMA-BB algorithm on randomly generated inputs of users channels and SIC capabilities.
- Simulation results from running the trained neural network with test data shows that it performs comparably to the heuristics it was trained from. This validates the proposed approach of training networks to make clustering decisions offline and then applying the trained model directly in live networks to make clustering decisions on a milisecond granularity.

C. PAPER ORGANIZATION

The rest of this paper is organized as follows. In Section II, the system model for the mmWave-NOMA system is presented and the overall user clustering and ordering problem is framed. Section III details the proposed neural network and how it is trained to solve the user clustering and ordering problem. Detailed simulation results for the proposed neural network, including a performance comparison with NOMA-MEC on the test data set are presented in Section IV. Finally, concluding remarks are provided in Section V.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we outline the system model used and the objective cluster minimization sub-problem to the overall rate maximization problem and the relevant details of the NOMA-MEC algorithm from [1] that the neural network uses to create the data sets that it learns from.

There are N single-antenna users, each with a minimum QoS constraint that need to be served. Each of these users have their own individual end user signal processing capabilities related to the SIC decoding procedure in NOMA that needs to be modelled. These N users are served by a single-cell BS having M antennas and operating in the mmWave spectrum; where the mmWave channel between the BS and user-u can be modelled in a similar way to [1] and [18] and is given as

$$\boldsymbol{h}_u = \boldsymbol{a}(\theta_u)\frac{\alpha_u}{\sqrt{L}\left(1+r_u^{\eta}\right)}, \tag{1}$$

where $\boldsymbol{a}(\theta_u)$ represents the steering vector, α_u represents the complex channel gain for user-u, L denotes the number of paths, r_u denotes the distance between the BS and user-u and

η denotes the path loss exponent. Further, like [1], analog beamforming (ABF) is used and only one beam can be transmitted at a time, which can be equated to forming one beam to serve one cluster of NOMA users per channel use, e.g., per time slice. Let the entire coverage region, $\bar{\theta}$, from $-\pi/2$ to $\pi/2$ be covered by a set of $B+1$ candidate beams that the BS can create through ABF. Thus, a NOMA cluster of users will be served on an orthogonal channel using one of these $B+1$ beams. Each beam-b in this list of candidate beams has the precoding vector,

$$\boldsymbol{w}_b = \boldsymbol{a}(\bar{\theta}_b), \quad \forall b \in [0, B], \tag{2}$$

where the parameter $\bar{\theta}_b$ is

$$\bar{\theta}_b = -\pi/2 + (b \times \pi/B). \tag{3}$$

Thus, there are a set of $B+1$ precoding vectors from (2) that cover the coverage region of $\bar{\theta} = -\pi/2$ to $\pi/2$, including some possible overlap between the beams, depending on the value of B. Using the same notations as [1], let B_c represent this list of candidate beams, such that B_c = {Beam-0, .., Beam-B}, with their respective list of candidate precoding vectors being $W_c = \{\boldsymbol{w}_0, .., \boldsymbol{w}_B\}$.

There are N users in the system and they can be clustered into K NOMA clusters, $K \leq N$. Each cluster is served by one of the $B+1$ precoding vectors in the candidate list in one orthogonal channel, e.g., time slice. Let $C = \{C_1, .., C_K\}$ represent the K clusters that are selected to collectively serve the N users in K time slices, where C_k refers to the N_{C_k} users selected to serve in the cluster with index-k, $N_{C_k} \leq N$. Let beam-b_k with a precoding vector $\boldsymbol{w}_{b_k}$ represent the beam selected for cluster C_k. In each cluster, C_k, the BS applies superposition coding (SC) as follows:

$$s_k = \sum_{u=1}^{N_{C_k}} \sqrt{p_u} s_{k,u}, \tag{4}$$

where p_u represents the power allocated to user-u. The received signal at user-u in cluster C_k is

$$\begin{aligned} y_u &= \boldsymbol{h}_u^H \boldsymbol{w}_{b_k} s_k + \xi_u, \\ &= \underbrace{\boldsymbol{h}_u^H \boldsymbol{w}_{b_k} \sqrt{p_u} s_{k,u}}_{\text{Desired signal}} + \underbrace{\boldsymbol{h}_u^H \boldsymbol{w}_{b_k} \sum_{u \neq v, v=1}^{n_k} \sqrt{p_v} s_{k,v}}_{\text{Inter-user interference}} + \underbrace{\xi_u}_{\text{Noise}}. \end{aligned} \tag{5}$$

We let $\pi_k(j)$ denote the user index for the j-th decoded user in the cluster C_k serving N_{C_k} users, $j \leq N_{C_k}$. The j-th user needs to first decode the signals of users $\{\pi_k(1), .., \pi_k(j)\}$ before decoding its own signal in the SIC procedure. When decoding user $\pi_k(j)$ at user $\pi_k(j')$, $j' > j$, the signal-to-interference-plus-noise ratio (SINR) can be represented as

$$\Gamma_{\pi_k(j')}^{\pi_k(j)} = \frac{p_j |\boldsymbol{h}_{(j')}^H \boldsymbol{w}_{b_k}|^2}{|\boldsymbol{h}_{(j')}^H \boldsymbol{w}_{b_k}|^2 \sum_{v>j}^{N_{C_k}} p_v + \sigma^2}, \tag{6}$$

where σ^2 is the noise power. If we let R_k denote the rate achieved in NOMA cluster C_k, the effective sum rate of the system, R_{sum} can be expressed as the sum of the rates from each cluster, R_k, given that each cluster is served by one channel. The effective sum rate can thus be represented as

$$\begin{aligned} R_{\text{sum}} &= \frac{\sum_{k=1}^{K} R_k}{K} \\ &= \frac{\sum_{k=1}^{K} \sum_{u \in C_k} \log_2\left(1 + \Gamma_{\pi_k(u)}^{\pi_k(u)}\right)}{K}, \end{aligned} \tag{7}$$

expressed in bits per second (bps) per channel-use.

The SIC decoding capability constraint of each user-u is modelled as d_u [1]. For example, $d_u = 4$ implies a user capable of decoding four other users' signals. This impacts the user ordering done within a cluster. If a user-u is placed in cluster-k at position j, then the maximum value of j is d_u. We let $d_{\max} = \max(d_u), \forall u = [1, .., N]$, represent the maximum decoding capability among the N users in the system. A heterogeneous system allows each user to have its own value of d_u, $d_u \leq d_{max}$; while a homogeneous system puts the additional constraint that all users have the same value of $d_u = d_{\max}$.

The objective optimization problem that needs to be solved is the same rate maximization problem as in [1], which we will instead tackle with the neural network approach in this paper and can be given as

$$\max_{\{C_k\},\{w_{b_k}\},\{\pi_k\},\{p_u\}} \quad R_{\text{sum}}, \tag{8a}$$

$$\text{s.t.} \quad R_u \geq \log_2(1 + \Gamma_{\min}), \quad \forall u = 1, .., N \tag{8b}$$

$$d_{\pi_k(j)} \geq j - 1, \quad j = 1, .., N_{C_k}, \ \forall k = 1, .., K, \tag{8c}$$

$$\sum_{i=1}^{N_{C_k}} p_i \leq P, \quad \forall k = 1, .., K, \tag{8d}$$

where $\Gamma_{\min}$ denotes the minimum SINR with which each user needs to be served, i.e., $\Gamma_{\pi_k(u)}^{\pi_k(u)} \geq \Gamma_{\min}$, $\forall u = [1, .., N]$, (8b) represents the QoS constraint, (8c) represents the decoding capability constraint, and (8d) represents the power per channel constraint. As in [1] the problem can be boken down into two sub-problems: (a) a joint user clustering, user ordering, and beamforming problem, and (b) a power allocation problem.

In this paper, the user clustering and ordering problem is what is relevant and can be given as

$$\min_{\{u_{i,k}\},\{b_k\},\{\pi_k\}} \quad K, \tag{9a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} u_{i,k} = 1, \quad \forall i = 1, .., N, \tag{9b}$$

$$b_k \in B_u, \ u_{i,k} = 1, \quad \forall k = 1, .., K, \tag{9c}$$

$$d_{\pi_k(j)} \geq j - 1, \quad j = 1, .., N_{C_k}, \ \forall k = 1, .., K, \tag{9d}$$

$$u_{i,k} \in \{0, 1\}, \tag{9e}$$

$$b_k \in B_c, \quad \forall k = 1, .., K, \tag{9f}$$

where (9b) makes sure a user is placed in exactly one cluster, (9c) assures that the beam chosen to serve a cluster of NOMA users belongs tois contained in the user-beam list of each of those users and (9d) is the key constraint that ensures the SIC decoding capability constraints of each user in the system is respected. It is worth noting that for homogeneous systems, constraint (9d) can be simplified down to

$$\sum_{i=1}^{N} u_{i,k} \leq d_{\max}, \quad \forall k = 1, .., K, \tag{10}$$

since all users have the same decoding capability and hence, only the overall number of users per cluster needs to be limited.

The objective function defined in (9a) represents an NP-complete problem, a class of problems widely known for its complexity in wireless communications and other fields of study. NOMA-MEC and NOMA-BB are two heuristics proposed to solve the problem in [1]. NOMA-MEC solves the cluster minimization problem in (9a) for a heterogeneous system while NOMA-BB is a lower complexity algorithm for the simpler homogeneous system. While we refer the reader to [1] for the details on the NOMA-MEC and NOMA-BB algorithm, these algorithms are used to create the training data set for the ANN-NOMA-MEC and ANN-NOMA-BB algorithms proposed in this paper. In other words, by generating datasets using the NOMA-MEC heuristic in a wide variety of simulation settings and knowing its performance in terms of R_{sum}, we establish the ground truth for the proposed artificial neural netowrks, i.e., the truth with which the neural networks are trained. We note that being heuristics, NOMA-MEC and NOMA-BB are approximations to the real solutions. However, the same neural network can just as easily be trained on a dataset created from the user clustering solution found from a brute-force style approach as well.

III. PROPOSED ALGORITHM(s)

In this section, we outline the proposed ANN-NOMA-MEC and ANN-NOMA-BB neural netowrks in terms of the neural network architecture in Section III-A and the detailed training and testing steps in Section III-B.

A. NEURAL NETWORK STRUCTURE

The proposed neural network architecture to search and learn underlying patterns in the datasets containing the user clustering solutions generated from the NOMA-MEC and NOMA-BB schemes is illustrated in Fig. 2. The neural network architecture consists of three fully connected layers: the input layer, a hidden layer and the output layer. The first layer is the input layer where the number of neurons is defined by the number of users N. Each neuron in this layer receives a data sample t containing a vector of features per user-u, $\boldsymbol{F}_u^{(t)}$. The relevant features of each user in $\boldsymbol{F}_u^{(t)}$ needed to train the neural network are defined by the user-beam set, $\boldsymbol{B}_u^{(t)}$, containing the best b beams for user-u; the users mmWave channel expressed through the physical

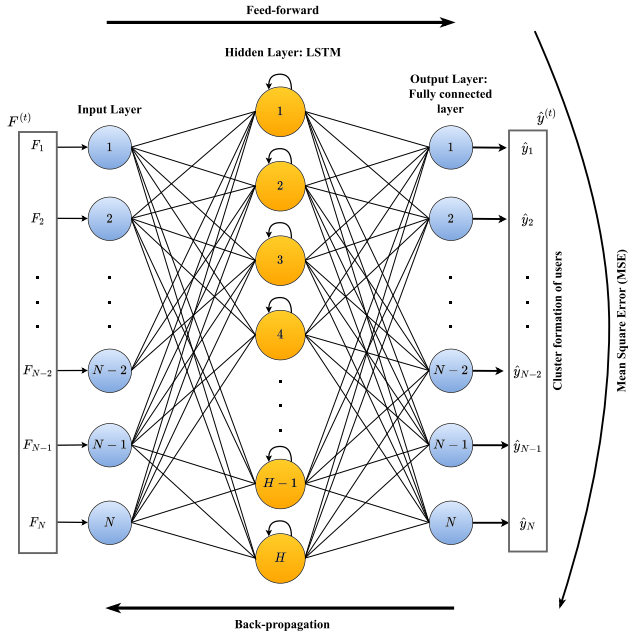


FIGURE 2. Proposed neural network structure for ANN-NOMA-MEC and ANN-NOMA-BB.

departure angle between the BS and the user-u, $\theta_u^{(t)}$, and the distance between the BS and the user-u, $r_u^{(t)}$; and finally the SIC decoding capability of user-u, $d_u^{(t)}$. Thus, the $\boldsymbol{F}_u^{(t)}$ user's feature vector will be a column vector with the following structure:

$$\boldsymbol{F}_u^{(t)} = \left[\boldsymbol{B}_u^{(t)}, \theta_u^{(t)}, r_u^{(t)}, d_u^{(t)}\right]^T. \tag{11}$$

Let n_f be the number of features contained in the feature vector. Finally, let $\boldsymbol{F}^{(t)}$ be the matrix of user features presented to the neural network at data sample t and is defined as

$$\boldsymbol{F}^{(t)} = \left[\boldsymbol{F}_1^{(t)}, \boldsymbol{F}_2^{(t)}, \cdots, \boldsymbol{F}_N^{(t)}\right]^T. \tag{12}$$

We can infer from (12) that the input to the neural network will be a matrix, whose dimension will be defined by the length of the feature vector, n_f, and the number of users N, i.e., $\boldsymbol{F}^{(t)} \in \mathbb{R}^{N \times n_f}$. It is worth noting that n_f can vary based on the number of beams per user set, b, in $\boldsymbol{B}_u$. For example, if $b = 2$ beams in $\boldsymbol{B}_u$, then $n_f = 5$ after adding the other features θ_u, r_u and d_u. Thus, each configuration of N and b will result in a different network size and, therefore, in a different neural network.

The second layer is the hidden layer consisting of H hidden neurons denoted by $h \in 1, 2, \cdots, H$. The input feature matrix, $\boldsymbol{F}^{(t)}$, containing the relevant user information from all input layer neurons ($i \in 1, 2, \cdots, N$) are connected to each neuron h present in the hidden layer. The selected hidden layer in our neural network architecture is the powerful long short-term memory (LSTM) layer, which makes each hidden neuron to be modeled as an LSTM unit. Fig. 3 illustrates in more detail the internal composition of hidden neurons and their connections with respect to the overall structure of the

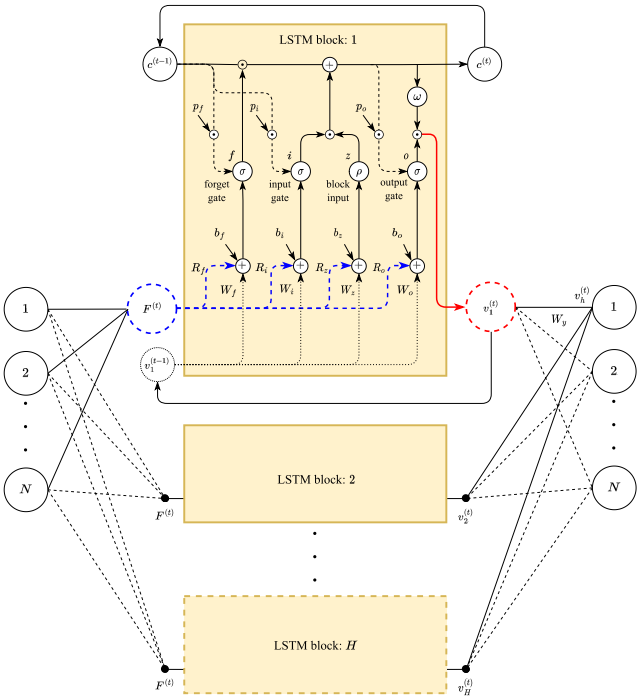


FIGURE 3. Illustrating the fully connected structure of the hidden neurons in the proposed neural network.

neural network as described later in this section. The output layer is the final layer and has N number of neurons denoted by $o \in 1, 2, \cdots, N$. The outputs from all neurons in the hidden layer are connected to each neuron o, which provides the estimated cluster assignment for the respective user-u, $\hat{y}_u$.

Our framework proposes a fully connected neural network structure. In the data sample t, each user-u feature vector $\boldsymbol{F}_u^{(t)}$ is received by each input layer neuron. The fully connected layout between the input and hidden layers allows the entire input feature matrix $\boldsymbol{F}^{(t)}$ to be processed by each hidden neuron. As shown in Fig. 3, hidden neurons are composed of a cell state denoted by $\boldsymbol{c}^{(t)}$ and four internal components defined by $\boldsymbol{z}^{(t)}$, $\boldsymbol{i}^{(t)}$, $\boldsymbol{f}^{(t)}$ and $\boldsymbol{o}^{(t)}$. These internal components manage the information flow by combining the input features $\boldsymbol{F}^{(t)}$ weighted by the weights $\boldsymbol{W}_z^{(t)}$, $\boldsymbol{W}_i^{(t)}$, $\boldsymbol{W}_f^{(t)}$, and $\boldsymbol{W}_o^{(t)}$, respectively; the hidden neuron outputs of the previous data sample $\boldsymbol{v}_h^{(t-1)}$ weighted by $\boldsymbol{R}_z^{(t)}$, $\boldsymbol{R}_i^{(t)}$, $\boldsymbol{R}_f^{(t)}$, and $\boldsymbol{R}_o^{(t)}$, respectively; and the biases $\boldsymbol{b}_z$, $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, and $\boldsymbol{b}_o$. Particularly, $\boldsymbol{f}^{(t)}$ and $\boldsymbol{i}^{(t)}$ add the cell state of the previous data sample $\boldsymbol{c}^{(t-1)}$ weighted by $\boldsymbol{p}_f^{(t)}$ and $\boldsymbol{p}_i^{(t)}$, respectively; while $\boldsymbol{o}^{(t)}$ adds the current cell state $\boldsymbol{c}^{(t)}$ weighted by $\boldsymbol{p}_o^{(t)}$. Then, the components $\boldsymbol{f}^{(t)}$, $\boldsymbol{i}^{(t)}$ and $\boldsymbol{o}^{(t)}$ are regulated by an activation function defined by σ while $\boldsymbol{z}^{(t)}$ is regulated by the activation ρ. Finally, the current cell state $\boldsymbol{c}^{(t)}$ is obtained from $\boldsymbol{c}^{(t-1)}$, $\boldsymbol{f}^{(t)}$, $\boldsymbol{i}^{(t)}$ and $\boldsymbol{z}^{(t)}$ while the output $\boldsymbol{v}_h^{(t)}$ is obtained from $\boldsymbol{o}^{(t)}$ and an regulated version of $\boldsymbol{c}^{(t)}$ through the activation function ω. Concretely, the idea behind the LSTM hidden layer is to maintain its state, $\boldsymbol{c}^{(t)}$, over time and regulate the flow of information through nonlinear activation functions (σ, ρ and ω). Thus, learning in the hidden neurons is given by recursively connecting their cell states, $\boldsymbol{c}^{(t-1)}$, and their outputs, $\boldsymbol{v}_h^{(t-1)}$, with their inputs $\boldsymbol{F}^{(t)}$. The above, gives rise to the STM and LTM concepts where: STM, short-term memory, refers to the learning at each data sample and is guided by the current output, $\boldsymbol{v}_h^{(t)}$; while LTM refers to the learning over time, which is recorded and remembered in the cell state $\boldsymbol{c}^{(t)}$. In this way, the hidden layer can remember relevant information and forget irrelevant information from user features, in favor of learning complex and unpredictable patterns to predict clustering solutions from the NOMA-MEC and NOMA-BB heuristics in the final layer.

The outputs of the hidden layer, $\boldsymbol{v}_h^{(t)} = v_1^{(t)}, v_2^{(t)}, \cdots, v_H^{(t)}$, are then passed through an activation function defined by φ, to obtain a regularized version, $\boldsymbol{h}^{(t)} = h_1^{(t)}, h_2^{(t)}, \cdots, h_H^{(t)}$, that feeds the final layer. The full connection between the hidden and output layers allows each neuron in the final layer to process $\boldsymbol{h}^{(t)}$ by a linear combination weighted by $\boldsymbol{W}_y^{(t)}$, whose final result is also regulated by the φ activation. In this way, the neural network obtains the estimated clustering assignment of each user-u, $\hat{y}_u$, as a result of a series of direct and indirect operations on the user features $\boldsymbol{F}^{(t)}$ where all the weights involved in the neural network structure we have proposed intervene: $\boldsymbol{W}_z^{(t)}$, $\boldsymbol{W}_i^{(t)}$, $\boldsymbol{W}_f^{(t)}$, $\boldsymbol{W}_o^{(t)}$, $\boldsymbol{R}_z^{(t)}$, $\boldsymbol{R}_i^{(t)}$, $\boldsymbol{R}_f^{(t)}$, $\boldsymbol{R}_o^{(t)}$, $\boldsymbol{p}_f^{(t)}$, $\boldsymbol{p}_i^{(f)}$, $\boldsymbol{p}_o^{(t)}$ and $\boldsymbol{W}_y^{(t)}$. As we will see in the next section, during the training phase, our neural network learns to estimate clustering solutions from the NOMA-MEC and NOMA-BB heuristics based on an iterative adjustment of these weights.

B. TRAINING AND TESTING

In this section we outline the training and testing phases for the proposed neural networks that applies to both ANN-NOMA-MEC and ANN-NOM-BB. Algorithm 1 summarizes the training phase in which our neural network will learn to cluster users based on the clustering solutions offered by the NOMA-MEC and NOMA-BB heuristics. Algorithm 2 summarises the testing phase by providing a method for evaluating the neural network as it is fed by user features to obtain the estimated clustering solution.

We begin with the training algorithm where we first need to describe the input defined by the sets $\boldsymbol{F}_{\mathcal{T}}$ and $\boldsymbol{Y}_{\mathcal{T}}$. Specifically, $\boldsymbol{F}_{\mathcal{T}} = \{\boldsymbol{F}^{(1)}, \boldsymbol{F}^{(2)}, \cdots, \boldsymbol{F}^{(\mathcal{T})}\}$ is a set containing the feature matrix $\boldsymbol{F}^{(t)}$ obtained from (12) for each data sample $t = 1, 2, \cdots, \mathcal{T}$. On the other hand, $\boldsymbol{Y}_{\mathcal{T}} = \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \cdots, \boldsymbol{y}^{(\mathcal{T})}\}$ is a set containing the labeled user clustering vector $\boldsymbol{y}^{(t)}$ for each data sample t. Let y_u be the actual cluster label of user-u, obtained from the clustering solution of the NOMA-MEC and NOMA-BB heuristics, the labeled vector following the structure $\boldsymbol{y}^{(t)} = [y_1, y_2, \cdots, y_u, \cdots, y_N]$, can be seen as the target clustering formation that the neural network must learn to predict. In other words, each $\boldsymbol{y}^{(t)}$ in $\boldsymbol{Y}_{\mathcal{T}}$ will be in charge of guiding the training of the neural network on each training data sample t as we will describe later in this section. As we see, the cluster numbering of each user-u in $\boldsymbol{y}^{(t)}$ plays

Algorithm 1 ANN Training

Input: $\boldsymbol{F}_{\mathcal{T}}$, $\boldsymbol{Y}_{\mathcal{T}}$
Parameters: α, n_{epochs}, H, $\mathcal{T}_{train}$, $\mathcal{T}_{valid}$
Output: ANN, the Artificial Neural Network model

Step-1: Split $\boldsymbol{F}_{\mathcal{T}}$, $\boldsymbol{Y}_{\mathcal{T}}$ using $\mathcal{T}_{train}$, $\mathcal{T}_{valid}$
– Get training sets $\boldsymbol{F}_{\mathcal{T}_{train}}$, $\boldsymbol{Y}_{\mathcal{T}_{train}}$
– Get validation sets $\boldsymbol{F}_{\mathcal{T}_{valid}}$, $\boldsymbol{Y}_{\mathcal{T}_{valid}}$
Step-2: Initializations
– Initialize $\boldsymbol{W}_{y}^{(0)}$, $\boldsymbol{W}_{*}^{(0)}$, $\boldsymbol{R}_{*}^{(0)}$, $\boldsymbol{p}_{i}^{(0)}$, $\boldsymbol{p}_{f}^{(0)}$, $\boldsymbol{p}_{o}^{(0)}$
– Initialize $\boldsymbol{y}^{(0)}$, $\boldsymbol{c}^{(0)}$
– Initialize bias $\boldsymbol{b}_{*}$
Step-3: Learning of the network
for $e = 1$ *to* n_{epochs} **do**
  for $t = 1$ *to* $\mathcal{T}_{train}$ **do**
    Feed-forward computation:
    – Vectorize $\boldsymbol{F}_{\mathcal{T}_{train}}^{(t)}$
    – Compute $\boldsymbol{z}^{(t)}$, $\boldsymbol{i}^{(t)}$, $\boldsymbol{f}^{(t)}$, $\boldsymbol{c}^{(t)}$, $\boldsymbol{o}^{(t)}$
    – Compute $\boldsymbol{v}_{h}^{(t)}$, $\boldsymbol{h}^{(t)}$
    – Compute $\boldsymbol{v}_{y}^{(t)}$, $\hat{\boldsymbol{y}}^{(t)}$
    – Compute the loss $\mathcal{L}$ using $\boldsymbol{Y}_{\mathcal{T}_{train}}^{(t)}$
    Back-propagation and weight adjustment:
    – Compute $\boldsymbol{\delta}_{v_y}^{(t)}$, $\boldsymbol{\Delta}_{v_h}^{(t)}$, $\boldsymbol{\delta}_{v_h}^{(t)}$
    – Compute $\boldsymbol{\delta}_{o}^{(t)}$, $\boldsymbol{\delta}_{c}^{(t)}$, $\boldsymbol{\delta}_{f}^{(t)}$, $\boldsymbol{\delta}_{i}^{(t)}$, $\boldsymbol{\delta}_{z}^{(t)}$
    – Compute $\boldsymbol{\delta}_{W_y}^{(t)}$, $\boldsymbol{\delta}_{W_*}^{(t)}$, $\boldsymbol{\delta}_{R_*}^{(t)}$, $\boldsymbol{\delta}_{p_i}^{(t)}$, $\boldsymbol{\delta}_{p_f}^{(t)}$, $\boldsymbol{\delta}_{p_o}^{(t)}$
    – Adjust $\boldsymbol{W}_{y}^{(t)}$, $\boldsymbol{W}_{*}^{(t)}$, $\boldsymbol{R}_{*}^{(t)}$, $\boldsymbol{p}_{i}^{(t)}$, $\boldsymbol{p}_{f}^{(t)}$, $\boldsymbol{p}_{o}^{(t)}$
  MSE computation using $\boldsymbol{Y}_{\mathcal{T}_{train}}$
Step-4: Validation
for $t = 1$ *to* $\mathcal{T}_{valid}$ **do**
  Feed-forward computation:
  – Vectorize $\boldsymbol{F}_{\mathcal{T}_{valid}}^{(t)}$
  – Compute $\boldsymbol{z}^{(t)}$, $\boldsymbol{i}^{(t)}$, $\boldsymbol{f}^{(t)}$, $\boldsymbol{c}^{(t)}$, $\boldsymbol{o}^{(t)}$
  – Compute $\boldsymbol{v}_{h}^{(t)}$, $\boldsymbol{h}^{(t)}$
  – Compute $\boldsymbol{v}_{y}^{(t)}$, $\hat{\boldsymbol{y}}_{valid}^{(t)}$
MSE computation using $\boldsymbol{Y}_{\mathcal{T}_{valid}}$

Algorithm 2 ANN Testing

Input: ANN, $\boldsymbol{F}_{\mathcal{T}_{test}}$, $\mathcal{T}_{test}$
Output: $\hat{\boldsymbol{Y}}_{\mathcal{T}_{test}} = \{\hat{\boldsymbol{y}}_{test}^{(1)}, \hat{\boldsymbol{y}}_{test}^{(2)}, \cdots, \hat{\boldsymbol{y}}_{test}^{(\mathcal{T}_{test})}\}$, the predicted cluster formation of users for all $\mathcal{T}_{test}$ data samples

Step-1: Initializations
– Initialize $\boldsymbol{y}^{(0)}$, $\boldsymbol{c}^{(0)}$
Step-2: Testing
for $t = 1$ *to* $\mathcal{T}_{test}$ **do**
  Feed-forward computation:
  – Vectorize $\boldsymbol{F}_{\mathcal{T}_{test}}^{(t)}$
  – Compute $\boldsymbol{z}^{(t)}$, $\boldsymbol{i}^{(t)}$, $\boldsymbol{f}^{(t)}$, $\boldsymbol{c}^{(t)}$ and $\boldsymbol{o}^{(t)}$
  – Compute $\boldsymbol{v}_{h}^{(t)}$, $\boldsymbol{h}^{(t)}$
  – Compute $\boldsymbol{v}_{y}^{(t)}$, $\boldsymbol{\mathcal{O}}_{test}^{(t)}$
  Store the predicted $\hat{\boldsymbol{y}}_{test}^{(t)}$:
  – $\hat{\boldsymbol{Y}}_{\mathcal{T}_{test}}\{t\} = \boldsymbol{\mathcal{O}}_{test}^{(t)}$

an important role and it should be labeled in a manner that can be easily learned by the neural network. Therefore, let $C = \{C_1, C_2, \cdots, C_k, \cdots, C_K\}$ be the set of K clusters and let $b_C = \{b_1, b_2, \cdots, b_k, \cdots, b_K\}$ be a set of K beams where each beam b_k represents the most repeated beam among all user-beam sets B_u involved with users in C_k, then, in $\boldsymbol{y}^{(t)}$ all users of a specific cluster C_k are labeled with a single number based the smallest beam in b_C. More explicitly, the users of the cluster containing the smallest beam in b_C will be labeled as 1, the users of the cluster containing the second smallest beam will be labeled as 2, and so on.

The goal of Algorithm 1 is to train a neural network based on the structure captured in Figs. 2 and 3 so that it learns to cluster users on its own. To do this, we break down the training algorithm into four steps. In Step-1, we obtain the training and validation sets by splitting the input sets $\boldsymbol{F}_{\mathcal{T}}$ and $\boldsymbol{Y}_{\mathcal{T}}$. The training sets denoted by $\boldsymbol{F}_{\mathcal{T}_{train}}$ and $\boldsymbol{Y}_{\mathcal{T}_{train}}$ are used to train the neural network and are obtained by selecting $\mathcal{T}_{train}$ number of data samples from $\boldsymbol{F}_{\mathcal{T}}$ and $\boldsymbol{Y}_{\mathcal{T}}$, respectively. The validation sets denoted by $\boldsymbol{F}_{\mathcal{T}_{valid}}$ and $\boldsymbol{Y}_{\mathcal{T}_{valid}}$ are used to assess the performance of the neural network after training and are obtained by selecting $\mathcal{T}_{valid}$ number of data samples from $\boldsymbol{F}_{\mathcal{T}}$ and $\boldsymbol{Y}_{\mathcal{T}}$, respectively. It should be noted that the data samples for each set can be selected randomly but with no overlap between each set. In Step-2, we initialize all the training variables that require initialization at time step $t = 0$ with zeroes or random values. This includes all the weights that compose the network, the output and cell state of the hidden layer denoted by $\boldsymbol{v}_{h}^{(0)}$ and $\boldsymbol{c}^{(0)}$, respectively as well as the biases involved in the four components of the hidden layer.

In Step-3, we perform the learning of the network based on the user features and the labeled user clustering formation of all data samples $t = 1, 2, \cdots, \mathcal{T}_{train}$ contained in the training sets $\boldsymbol{F}_{\mathcal{T}_{train}}$ and $\boldsymbol{Y}_{\mathcal{T}_{train}}$, respectively. Let e be a training epoch in which the network is learning to estimate the desired user clustering formations in $\boldsymbol{Y}_{\mathcal{T}_{train}}$ from the corresponding user features in $\boldsymbol{F}_{\mathcal{T}_{train}}$, the training algorithm needs to go through several epochs until it ensures that the estimated clustering formation of the network is close enough to the desired ones. The learning at each epoch e is performed by iterating through the sets $\boldsymbol{F}_{\mathcal{T}_{train}}$ and $\boldsymbol{Y}_{\mathcal{T}_{train}}$ where each iterative step is defined by the data sample t. Let $\boldsymbol{F}_{\mathcal{T}_{train}}^{(t)}$ and $\boldsymbol{Y}_{\mathcal{T}_{train}}^{(t)}$ be the user feature matrix and the labeled clustering vector of the data sample t from $\boldsymbol{F}_{\mathcal{T}_{train}}$ and $\boldsymbol{Y}_{\mathcal{T}_{train}}$, respectively, we break down the learning in two stages. In the first stage the feed-forward computation of the neural network is performed starting from the feature matrix $\boldsymbol{F}_{\mathcal{T}_{train}}^{(t)}$ to finish computing the network loss

$\mathcal{L}$ using the labeled clustering vector $\boldsymbol{Y}^{(t)}_{\mathcal{T}_{train}}$. Then, in the second stage the back-propagation is carried out to adjust all the weights of the neural network where its learning is reflected.

The feed-forward computation begins by vectorizing the user feature matrix $\boldsymbol{F}^{(t)}_{\mathcal{T}_{train}}$ to convert it into a column vector. The vectorization of the $\boldsymbol{F}^{(t)}_{\mathcal{T}_{train}}$ matrix is obtained by stacking all its columns on top of one another. Let $\boldsymbol{F}^{(t)}_{v}$ be the vectorized version of $\boldsymbol{F}^{(t)}_{\mathcal{T}_{train}}$ defined as,

$$\begin{aligned}\boldsymbol{F}^{(t)}_{v} &= \text{vec}\left(\boldsymbol{F}^{(t)}_{\mathcal{T}_{train}}\right)\\ &= \Big[B^{(t)}_1, \cdots, B^{(t)}_N, \theta^{(t)}_1, \cdots, \theta^{(t)}_N,\\ &\qquad r^{(t)}_1, \cdots, r^{(t)}_N, d^{(t)}_1, \cdots, d^{(t)}_N\Big]^T. \end{aligned} \tag{13}$$

We can infer from (13) that $\boldsymbol{F}^{(t)}_{v} \in \mathbb{R}^{N_f \times 1}$ is a column vector containing the features of all users and whose length N_f is defined by the number of elements in $\boldsymbol{F}^{(t)}_{\mathcal{T}_{train}}$, i.e., $N_f = N \cdot n_f$.

Next, the feed-forward uses the vectorized version $\boldsymbol{F}^{(t)}_{v}$ to compute the four internal components, $\{z, i, f, o\}$,along with the cell state of the hidden layer as outlined in Fig. 3, as follows [28]:

$$\boldsymbol{z}^{(t)} = \rho\left(\boldsymbol{W}^{(t)}_{z}\boldsymbol{F}^{(t)}_{v} + \boldsymbol{R}^{(t)}_{z}\boldsymbol{v}^{(t-1)}_{h} + \boldsymbol{b}_z\right), \tag{14}$$

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}^{(t)}_{i}\boldsymbol{F}^{(t)}_{v} + \boldsymbol{R}^{(t)}_{i}\boldsymbol{v}^{(t-1)}_{h} + \boldsymbol{p}^{(t)}_{i} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{b}_i\right), \tag{15}$$

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}^{(t)}_{f}\boldsymbol{F}^{(t)}_{v} + \boldsymbol{R}^{(t)}_{f}\boldsymbol{v}^{(t-1)}_{h} + \boldsymbol{p}^{(t)}_{f} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{b}_f\right), \tag{16}$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{z}^{(t)} \odot \boldsymbol{i}^{(t)} + \boldsymbol{c}^{(t-1)} \odot \boldsymbol{f}^{(t)}, \tag{17}$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}^{(t)}_{o}\boldsymbol{F}^{(t)}_{v} + \boldsymbol{R}^{(t)}_{o}\boldsymbol{v}^{(t-1)}_{h} + \boldsymbol{p}^{(t)}_{o} \odot \boldsymbol{c}^{(t)} + \boldsymbol{b}_o\right). \tag{18}$$

Let $*$ refer to any of the internal components of the hidden layer $\{z, i, f, o\}$, then $\boldsymbol{W}^{(t)}_{*} \in \mathbb{R}^{H \times N_f}$, $\boldsymbol{R}^{(t)}_{*} \in \mathbb{R}^{H \times H}$, $\boldsymbol{b}_{*} \in \mathbb{R}^{H \times 1}$, $*^{(t)} \in \mathbb{R}^{H \times 1}$. The operator $\odot$ denotes the pointwise multiplication between two vectors that must necessarily have the same length. For example, in expression (17) the operation $\boldsymbol{z}^{(t)} \odot \boldsymbol{i}^{(t)}$ is performed, for which the block input $\boldsymbol{z}^{(t)} \in \mathbb{R}^{H \times 1}$ and the input gate $\boldsymbol{i}^{(t)} \in \mathbb{R}^{H \times 1}$ are two column vectors of length H. The result of this operation is another column vector of length H in which each element reflects the multiplication of the corresponding elements in $\boldsymbol{z}^{(t)}$ and $\boldsymbol{i}^{(t)}$. Thus, we can deduce that the cell state $\boldsymbol{c}^{(t)} \in \mathbb{R}^{H \times 1}$ in which each element is associated to each hidden neuron $h = 1, \cdots, H$. It is worth highlighting the recurrence of the second term in the definition of the four internal components of the hidden layer, $\boldsymbol{v}^{(t-1)}_{h} \in \mathbb{R}^{H \times 1}$, that represents the output of the hidden layer in a previous iterative step. During the first iteration, $t = 1$, $\boldsymbol{v}^{(0)}_{h}$ which was initialized in Step-2 is applied.

Then, both the output of the hidden layer $\boldsymbol{v}^{(t)}_{h}$ and its regularised version $\boldsymbol{h}^{(t)}$ are computed by applying the following expressions,

$$\boldsymbol{v}^{(t)}_{h} = \omega\left(\boldsymbol{c}^{(t)}\right) \odot \boldsymbol{o}^{(t)}, \tag{19}$$

$$\boldsymbol{h}^{(t)} = \varphi\left(\boldsymbol{v}^{(t)}_{h}\right). \tag{20}$$

We can infer from (19) that $\boldsymbol{v}^{(t)}_{h} \in \mathbb{R}^{H \times 1}$ is a vector of length H in which each element represents the output of each hidden neuron $h = 1, \cdots, H$. While $\boldsymbol{h}^{(t)}$ is simply the result of passing $\boldsymbol{v}^{(t)}_{h}$ through the activation function φ. Similar to [14], the activation φ that we apply here is the ReLu function for which each element in $\boldsymbol{v}^{(t)}_{h}$ is regulated depending on its sign. Elements with positive sign are treated linearly, while elements with negative sign are set to be zero. In this way, $\boldsymbol{h}^{(t)} \in \mathbb{R}^{H \times 1}$ results in a column vector of length H where each regulated element is associated to each neuron $h = 1, 2, \cdots, H$.

Subsequently, both the output of the final layer $\boldsymbol{v}^{(t)}_{y}$ and the estimated user clustering formation $\hat{\boldsymbol{y}}^{(t)}$ are computed following the expressions,

$$\boldsymbol{v}^{(t)}_{y} = \boldsymbol{W}^{(t)}_{y}\boldsymbol{h}^{(t)}, \tag{21}$$

$$\hat{\boldsymbol{y}}^{(t)} = \varphi\left(\boldsymbol{v}^{(t)}_{y}\right). \tag{22}$$

In (21), $\boldsymbol{W}^{(t)}_{y} \in \mathbb{R}^{N \times H}$ represents a weight matrix containing all the weights associated with the output layer. Specifically, each row in $\boldsymbol{W}^{(t)}_{y}$ is associated with each output neuron $o = 1, 2, \cdots, N$. Since each of these neurons is fed with the vector $\boldsymbol{h}^{(t)}$ the matrix multiplication defined in (21) results in the vector $\boldsymbol{v}^{(t)}_{y} \in \mathbb{R}^{N \times 1}$ in which each element is related to a neuron o. Similar to (20), in (22) $\boldsymbol{v}^{(t)}_{y}$ is passed through the ReLu activation function to obtain the estimated user clustering formation $\hat{\boldsymbol{y}}^{(t)} \in \mathbb{R}^{N \times 1}$ as a vector of length N, in which each element represents the estimated cluster assignment of the user-u, $\hat{y}^{(t)}_{u}$, for each $u = 1, 2, \cdots, N$.

Finally, feed-forward stage computes the network loss denoted by $\mathcal{L}$ in which the estimated clustering formation $\hat{\boldsymbol{y}}^{(t)}$ is compared with the labeled clustering formation $\boldsymbol{Y}^{(t)}_{\mathcal{T}_{train}}$ by the following expression,

$$\mathcal{L} = \frac{1}{N}\sum_{u=1}^{N}\left(y_u - \hat{y}_u\right)^2. \tag{23}$$

The back-propagation stage seeks to adjust the learnable parameters involved in the network based on the minimization of the network loss $\mathcal{L}$ obtained in the previous stage. As pointed out in [28], at each iterative step t the back-propagation begins by applying the generalized delta rule to get the delta vectors $\boldsymbol{\delta}^{(t)}_{v_y}$ and $\boldsymbol{\Delta}^{(t)}_{v_h}$ that enable the update of weights for the output and hidden layers, respectively, and can be expressed as

$$\boldsymbol{\delta}^{(t)}_{v_y} = \varphi'\left(\boldsymbol{v}^{(t)}_{y}\right) \odot \boldsymbol{e}'^{(t)}_{y}, \tag{24}$$

$$\boldsymbol{\Delta}^{(t)}_{v_h} = \varphi'\left(\boldsymbol{v}^{(t)}_{h}\right) \odot \boldsymbol{e}'^{(t)}_{h}, \tag{25}$$

where φ' represents the derivative of the ReLu activation function upon $\boldsymbol{v}^{(t)}_{y}$ and $\boldsymbol{v}^{(t)}_{h}$. On the other hand, $\boldsymbol{e}'^{(t)}_{y}$ represents

the derivative of the network loss while $\boldsymbol{e}_h^{\prime(t)} = (\boldsymbol{W}_y)^T \boldsymbol{\delta}_{v_y}^{(t)}$. Next, we compute the accumulated gradient vector $\boldsymbol{\delta}_{v_h}^{(t)}$, which can be defined as the combination between $\boldsymbol{\Delta}_{v_h}^{(t)}$ and the recurrent dependencies according to the expression,

$$\begin{aligned}\boldsymbol{\delta}_{v_h}^{(t)} = \boldsymbol{\Delta}_{v_h}^{(t)} + \boldsymbol{R}_z^{(t-1)}\boldsymbol{\delta}_z^{(t+1)} + \boldsymbol{R}_i^{(t-1)}\boldsymbol{\delta}_i^{(t+1)} \\ + \boldsymbol{R}_f^{(t-1)}\boldsymbol{\delta}_f^{(t+1)} + \boldsymbol{R}_o^{(t-1)}\boldsymbol{\delta}_o^{(t+1)},\end{aligned} \tag{26}$$

such that $\boldsymbol{R}_*^{(t-1)}$ (where $*$ can be z, i, f, o) represent recurrent weights that were adjusted at the previous iterative step; while $\boldsymbol{\delta}_*^{(t+1)}$ represent the gradient vectors associated with the four components of the hidden layer at $t+1$. Subsequently, we compute the delta vectors $\boldsymbol{\delta}_o^{(t)}$, $\boldsymbol{\delta}_c^{(t)}$, $\boldsymbol{\delta}_f^{(t)}$, $\boldsymbol{\delta}_i^{(t)}$, and $\boldsymbol{\delta}_z^{(t)}$ according to,

$$\begin{aligned}\boldsymbol{\delta}_o^{(t)} &= \boldsymbol{\delta}_{v_h}^{(t)} \odot \omega\left(\boldsymbol{c}^{(t)}\right) \odot \sigma'\left(\hat{\boldsymbol{o}}^{(t)}\right), \\ \boldsymbol{\delta}_c^{(t)} &= \boldsymbol{\delta}_{v_h}^{(t)} \odot \boldsymbol{o}^{(t)} \odot \omega'\left(\boldsymbol{c}^{(t)}\right) + \boldsymbol{p}_o \odot \boldsymbol{\delta}_o^{(t)} \\ &\quad + \boldsymbol{p}_i \odot \boldsymbol{\delta}_i^{(t+1)} + \boldsymbol{p}_f \odot \boldsymbol{\delta}_f^{(t+1)} + \boldsymbol{\delta}_c^{(t+1)} \odot \boldsymbol{f}^{(t+1)}, \\ \boldsymbol{\delta}_f^{(t)} &= \boldsymbol{\delta}_c^{(t)} \odot \boldsymbol{c}^{(t-1)} \odot \sigma'(\hat{\boldsymbol{f}}^{(t)}), \\ \boldsymbol{\delta}_i^{(t)} &= \boldsymbol{\delta}_c^{(t)} \odot \boldsymbol{z}^{(t)} \odot \sigma'(\hat{\boldsymbol{i}}^{(t)}), \\ \boldsymbol{\delta}_z^{(t)} &= \boldsymbol{\delta}_c^{(t)} \odot \boldsymbol{i}^{(t)} \odot \rho'(\hat{\boldsymbol{z}}^{(t)}),\end{aligned}$$

where $\hat{\boldsymbol{i}}^{(t)}$, $\hat{\boldsymbol{f}}^{(t)}$ and $\hat{\boldsymbol{o}}^{(t)}$ are vectors defined from (15), (16) and (18), respectively, but without applying the activation function σ; $\hat{\boldsymbol{z}}^{(t)}$ is a vector defined from (14) but without applying the activation function ρ; while ρ', ω' and σ' are the derivatives of the activation functions tanh and sigmoid, respectively.

Based on the computed delta vectors, the gradients $\boldsymbol{\delta}_{W_y}^{(t)}$, $\boldsymbol{\delta}_{W_*}^{(t)}$, $\boldsymbol{\delta}_{R_*}^{(t)}$, $\boldsymbol{\delta}_{p_i}^{(t)}$, $\boldsymbol{\delta}_{p_f}^{(t)}$ and $\boldsymbol{\delta}_{p_o}^{(t)}$ used to adjust the weights are calculated as follows:

$$\begin{aligned}\boldsymbol{\delta}_{W_y}^{(t)} &= \boldsymbol{\delta}_{v_y}^{(t)} \otimes \boldsymbol{h}^{(t)}, \quad \boldsymbol{\delta}_{p_i}^{(t)} = \boldsymbol{c}^{(t)} \odot \boldsymbol{\delta}_i^{(t+1)}, \\ \boldsymbol{\delta}_{W_*}^{(t)} &= \boldsymbol{\delta}_*^{(t)} \otimes \boldsymbol{F}_v^{(t)}, \quad \boldsymbol{\delta}_{p_f}^{(t)} = \boldsymbol{c}^{(t)} \odot \boldsymbol{\delta}_f^{(t+1)}, \\ \boldsymbol{\delta}_{R_*}^{(t)} &= \boldsymbol{\delta}_*^{(t+1)} \otimes \boldsymbol{v}_h^{(t)}, \quad \boldsymbol{\delta}_{p_o}^{(t)} = \boldsymbol{c}^{(t)} \odot \boldsymbol{\delta}_o^{(t)},\end{aligned}$$

with $*$ being any of the four internal components in the hidden layer $\{z, i, f, o\}$ and the operator $\otimes$ representing the outer product between two vectors. Unlike pointwise multiplication, the outer product $\otimes$ performs a matrix computation in which the second vector is transposed. For example, if we wish to compute the gradient $\boldsymbol{\delta}_{W_f}^{(t)} = \boldsymbol{\delta}_f^{(t)} \otimes \boldsymbol{F}_v^{(t)} = \boldsymbol{\delta}_f^{(t)} \left[\boldsymbol{F}_v^{(t)}\right]^T$, for which if $\boldsymbol{\delta}_f^{(t)} \in \mathbb{R}^{H\times 1}$ and $\left[\boldsymbol{F}_v^{(t)}\right]^T \in \mathbb{R}^{1\times N_f}$ the result of this operation is the matrix $\boldsymbol{\delta}_{W_f}^{(t)} \in \mathbb{R}^{H\times N_f}$.

Finally, similar to the work in [14], we apply the stochastic gradient descent (SGD) scheme in order to perform the adjustment of all the weights associated with our neural network. Mathematically, the adjustment of the weights can be achieved by following the expressions,

$$\begin{aligned}\boldsymbol{W}_y^{(t)} &= \boldsymbol{W}_y^{(t-1)} + \alpha\boldsymbol{\delta}_{W_y}^{(t)}, \quad \boldsymbol{p}_i^{(t)} = \boldsymbol{p}_i^{(t-1)} + \alpha\boldsymbol{\delta}_{p_i}^{(t)}, \\ \boldsymbol{W}_*^{(t)} &= \boldsymbol{W}_*^{(t-1)} + \alpha\boldsymbol{\delta}_{W_*}^{(t)}, \quad \boldsymbol{p}_f^{(t)} = \boldsymbol{p}_f^{(t-1)} + \alpha\boldsymbol{\delta}_{p_f}^{(t)}, \\ \boldsymbol{R}_*^{(t)} &= \boldsymbol{R}_*^{(t-1)} + \alpha\boldsymbol{\delta}_{R_*}^{(t)}, \quad \boldsymbol{p}_o^{(t)} = \boldsymbol{p}_o^{(t-1)} + \alpha\boldsymbol{\delta}_{p_o}^{(t)},\end{aligned}$$

where α is a high-impact scalar parameter in the training of the network which defines the learning rate that regulates how much the weights are adjusted at each iterative step. At the end of step-3, the mean square error (MSE) metric is computed which measures the performance of the network at the end of each epoch and is defined as,

$$\text{MSE} = \frac{1}{\mathcal{T}_{train}} \sum_{t=1}^{\mathcal{T}_{train}} \left(\boldsymbol{Y}_{\mathcal{T}_{train}}^{(t)} - \hat{\boldsymbol{y}}^{(t)}\right)^2. \tag{27}$$

In Step-4 of Algorithm 1, the performance is assessed at the end of each epoch e using the validation sets, $\boldsymbol{F}_{\mathcal{T}_{valid}}$ and $\boldsymbol{Y}_{\mathcal{T}_{valid}}$, whose data samples have not been used during learning and, therefore, helps measure the generalization capability of the network. To do this, we iteratively go through $\boldsymbol{F}_{\mathcal{T}_{valid}}$ and $\boldsymbol{Y}_{\mathcal{T}_{valid}}$ running the feed-forward stage described in Step-3 but using $\boldsymbol{F}_{\mathcal{T}_{valid}}^{(t)}$ at each iterative step in order to obtain the corresponding estimated clustering formation $\hat{\boldsymbol{y}}_{valid}^{(t)}$. Finally, we compute the MSE metric using $\hat{\boldsymbol{y}}_{valid}^{(t)}$ and $\boldsymbol{Y}_{\mathcal{T}_{valid}}^{(t)}$ to measure the validation performance. If the validation MSE is relatively high, then the learning needs to go through more epochs until the validation MSE reaches an acceptable performance.

The training algorithm is highly influenced by the parameters $\mathcal{T}_{train}$, α, H and n_{epochs}. In particular, $\mathcal{T}_{train}$ defines the number of training data samples selected from $\boldsymbol{F}_{\mathcal{T}}$ and $\boldsymbol{Y}_{\mathcal{T}}$ that are used for network learning. If $\mathcal{T}_{train}$ is too small the learning of the network in Step-3 can be highly compromised because the weights would be adjusted for a small set of solutions from the NOMA-MEC and NOMA-BB heuristics, as the case may be. This would reduce the generalization capability of the network, resulting in a low validation MSE. The learning rate α controls how much the network weights are adjusted during Step-3 at each iterative step t. In other words, this parameter defines how fast the network learns. Considering that α can take values between 0 and 1, a desired value of α should be low enough for the network to converge to something useful, but high enough that it can be trained in a reasonable time. In this sense, a smaller value of α requires more training epochs since only small changes are made to the weights at each adjustment; while a large value of α results in rapid changes and requires fewer training epochs but can lead to a sub-optimal set of weights. On the other hand, the design parameter H directly influences the complexity of the training algorithm. A large value of H leads to an exaggerated number of neurons in the hidden layer, making the neural network structure very large and therefore very slow. However, if H is too small, it reduces the ability of the hidden layer to learn to remember desired patterns as well as to forget undesired patterns. Finally, the parameter

n_{epochs} which defines the number of times the learning works through the complete $\boldsymbol{F}_{\mathcal{T}_{train}}$ and $\boldsymbol{Y}_{\mathcal{T}_{train}}$ sets, is a critical parameter as it ensures the convergence of the network to an acceptable performance. A desirable value of n_{epochs} should be large enough so that the train MSE is low enough while the validation MSE is not affected. Therefore, the design of the neural network relies on an appropriate choice of these four parameters in order to find an optimal combination that leads to a high performing network that can be trained in an acceptable amont of time. In Section IV, we illustrate the tuning process of these four parameters for the proposed neural netowrk.

In the testing algorithm described in Algorithm 2, the input is defined by the artificial neural network ANN and the testing set $\boldsymbol{F}_{\mathcal{T}_{test}}$. The ANN is a network that has been trained through Algorithm 1 for which all weights involved have been optimally adjusted. The $\boldsymbol{F}_{\mathcal{T}_{test}}$ set containing $\mathcal{T}_{test}$ feature matrices from $\boldsymbol{F}_{\mathcal{T}}$ is used to test the ANN in order to obtain the corresponding estimated clustering formations returned in $\hat{\boldsymbol{Y}}_{\mathcal{T}_{test}}$. To do this, we break down Algorithm 2 into two steps. In Step-1 we simply need to initialize the output and cell state of the hidden layer denoted by $\boldsymbol{y}^{(0)}$ and $\boldsymbol{c}^{(0)}$, respectively, to consider that these have initial values when $t = 0$. The initial values can be zeros. In Step-2, we perform ANN testing by iterating through $\boldsymbol{F}_{\mathcal{T}_{test}}$ where, at each iterative step $t = 1, 2, \cdots, \mathcal{T}_{test}$, the feed-forward stage described in Algorithm 1 is performed based on the feature matrix $\boldsymbol{F}^{(t)}_{\mathcal{T}_{test}}$ from $\boldsymbol{F}_{\mathcal{T}_{test}}$ to obtain the estimated clustering formation $\hat{\boldsymbol{y}}^{(t)}_{test}$. Then each $\hat{\boldsymbol{y}}^{(t)}_{test}$ is orderly stored in the set $\hat{\boldsymbol{Y}}_{\mathcal{T}_{test}}$ to ensure that the output of the testing algorithm contains all the test solutions estimated by the ANN.

IV. SIMULATION RESULTS AND DISCUSSION

In this section, we illustrate simulation results to show the process of fine-tuning the neural network training algorithm parameters and then compare the performance of the network against the heuristics from which it was trained. We work with a total dataset of $\mathcal{T} = 12000$ samples obtained by running the NOMA-MEC and NOMA-BB heuristics and unless otherwise stated, the number of training samples used is 70%of the data set, i.e., $\mathcal{T}_{train} = 8400$. Table 2 shows the simulation settings used in this section. For the simulations related to the tuning of neural netowrk parameters, we present the ANN-NOMA-MEC case as it is the more complicated network given that it supports heterogeneous systems where each user has its own individual SIC decoding capability presented to the network to factor into the clustering algorithm. By going through different controlled settings of hidden neurons, H, learning rate, α, training data samples, $\mathcal{T}_{train}$ and number of epochs, n_{epochs}, we train different neural networks with the objective of choosing the combination of parameters that optimizes the validation performance of the ANNs. In particular, to measure the validation performance, we use both the MSE metric and the effective sum rate R_{sum} [1] in the simulation runs. The MSE metric introduced in Section III, is chosen because it is commonly used to measure the performance of an ANN model, while R_{sum} provides us with a performance measure in terms of the user clustering formation predicted by the ANN. We also run simulations for different network sizes, i.e., using different configurations of number of users N and number of best b beams per user, to demonstrate the stability of the proposed neural network structure in ANN-NOMA-MEC and ANN-NOMA-BB, respectively. Finally, the effective sum rate for our optimally trained ANN-NOMA-MEC and ANN-NOMA-BB networks are benchmarked with the original NOMA-MEC and NOMA-BB heuristics, respectively.

TABLE 2. Simulation parameters.

Parameter name, notation	Value
Number of users , N	$[50, 100, 150, 200]$
Number of best beams, b	$[1, 2, 3, 4]$
Number of hidden neurons, H	$[1, 10, 50, 100, 150, 200]$
Learning Rate, α	$[0.01, 0.001]$
Training samples, $\mathcal{T}_{train}$	$[6000, 7200, 8400]$
Validation samples, $\mathcal{T}_{valid}$	1800
Testing samples, $\mathcal{T}_{test}$	1800
Number of epochs, n_{epochs}	20

We start by investigating the effect of the number of hidden neurons, H, on the performance of the proposed neural network as the number of users in the network grows. In Fig. 4, we run simulations for different choices of H for an ANN-NOMA-MEC neural network trained on a dataset of 8400 samples obtained from NOMA-MEC heuristic with $b = 2$. As seen in Fig. 4, in general, an ANN-NOMA-MEC setting with $H = 1$, i.e., a single hidden neuron, leads to the highest MSE which implies the lowest performance, because the clustering formations predicted by the network are very poor and are further away from the clustering formations solved by the NOMA-MEC heuristic. At $H = 10$, we get good performance at 50 users but the performance deteriorates as the number of users is further increased. In general, when $H = 50$ we observe the best MSE performance for all users. However, when we go to even higher number of hidden layer neurons, the neural netowrk starts to overfit to the training data and the performance deteriorates when tested on the validation dataset. Therefore, we use $H = 50$ hidden neurons in the further simulations.

We now move to study the impact of the learning rate, α, on the performance of ANN-NOMA-MEC. To do this, in Fig. 5, we compare both the MSE performance (left axis) and the effective sum rate R_{sum} (right axis) measured by the training algorithm when using learning rates of 0.001 and 0.01, while keeping the other training parameters fixed with $H = 50$ and $b = 2$. It is worth noting that a lower MSE and a higher value of R_{sum} are indicators of better performance. It is clear from the results in Fig. 5a and Fig. 5b, which are

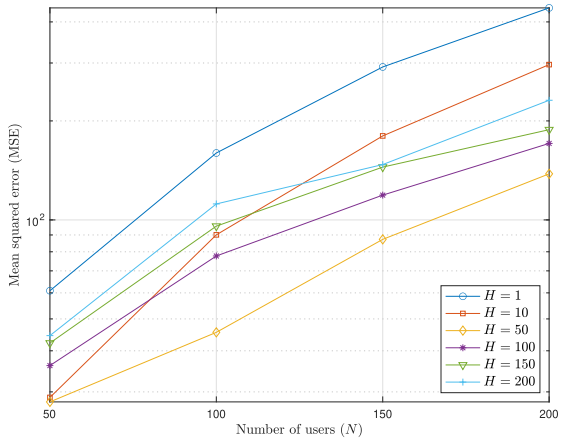


FIGURE 4. Effect of the number of neurons in the hidden layer, H, on the performance of ANN-NOMA-MEC for different numbers of users, N, with $b = 2$.

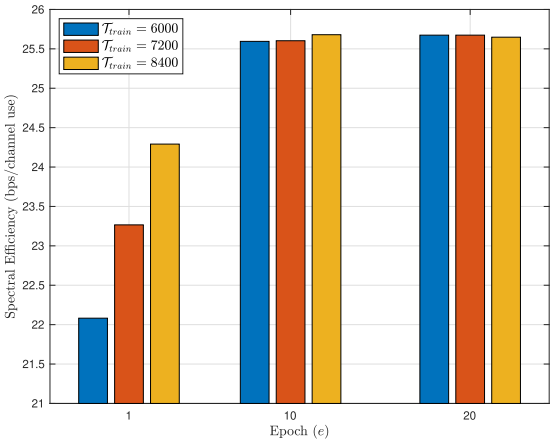
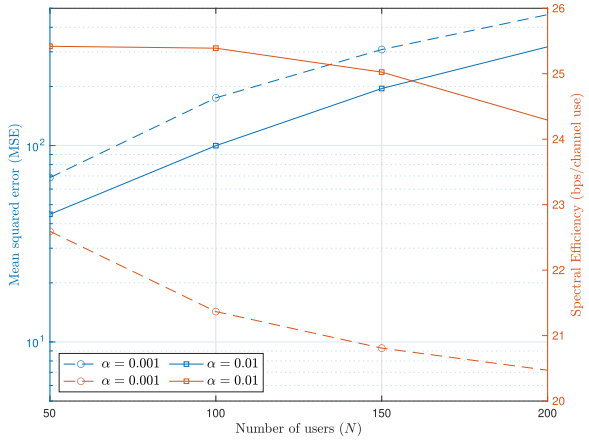
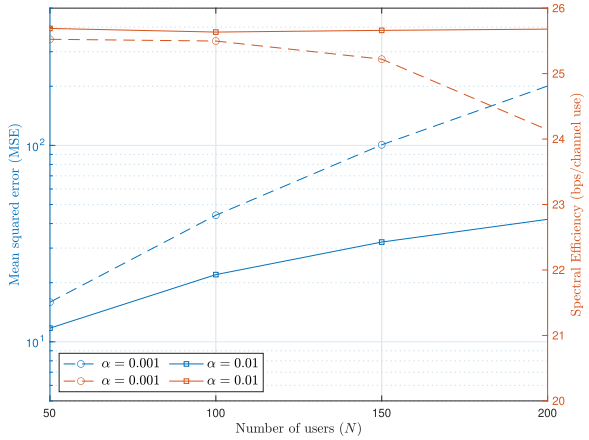


FIGURE 6. R_{sum} at different learning epochs, e, and different training data samples, $\mathcal{T}_{train}$. ANN-NOMA-MEC with $N = 200$ users and $b = 2$.



(a)



(b)

FIGURE 5. Impact of the learning rate, α, on MSE and R_{sum} for different numbers of users, N, at (a) $n_{epochs} = 1$ and (b) $n_{epochs} = 10$. ANN-NOMA-MEC with $b = 2$.

presented at $n_{epochs} = 1$ and $n_{epochs} = 10$, respectively, that at $\alpha = 0.01$, the training algorithm finds it easier to control the learning of the ANN-NOMA-MEC network, which leads to better MSE performances that translate into better estimates of clustering formations and, therefore, this leads to higher effective sum rates. Hence, we use $\alpha = 0.01$ as the learning rate in the simulations that follow.

Through the simulations in Fig. 6 and Fig. 7, we illustarte the impact of the parameters n_{epochs} and $\mathcal{T}_{train}$ on the performance of ANN-NOMA-MEC during learning. In Fig. 6, we can observe that setting the training algorithm with $n_{epochs} = 1$ is inadequate for any value of $\mathcal{T}_{train}$, which implies that the learning must go through more than one epoch for the ANN to better adjust its weights in order to achieve better estimates of clustering formations. The results between $n_{epochs} = 10$ and $n_{epochs} = 20$ are comparable for all values of $\mathcal{T}_{train}$, which suggests that the network has reached its learning peak at around the tenth epoch. At $n_{epochs} = 10$, we see that with $\mathcal{T}_{train} = 8400$, i.e., 70% of the total samples, we achieve the highest performance. We illustrate this further with the simulation results in Fig. 7, where we compare both the MSE performance (left axis) and the effective sum rate R_{sum} (right axis) measured by the training algorithm when considering 6400, 7200 and 8400 training data samples, while fixing the other parameters according to $H = 50$, $\alpha = 0.01$ and $n_{epochs} = 10$. From Fig. 7, it is clear that $\mathcal{T}_{train} = 6400$ is an insufficient number of samples for the network to train with for optimal performance. At $\mathcal{T}_{train} = 8400$ training samples, we find the best network performances, with the lowest validation MSEs, leading to the highest R_{sum}. Therefore, $\mathcal{T}_{train} = 8400$ training samples are used in the next set of simulations.

In this way, the parameters of ANN-NOMA-MEC and ANN-NOMA-BB can be fine-tuned and for further simulations, we set the training algorithm to work with $H = 50$ hidden neurons, a learning rate $\alpha = 0.01$, $\mathcal{T}_{train} = 8400$ training data samples and $n_{epochs} = 10$ training epochs. Using this combination of parameters, we can train ANN-NOMA-MEC and ANN-NOMA-BB offline from a dataset created by the NOMA-MEC and NOMA-BB parameters, respectively, and then apply them to make clustering decisions in

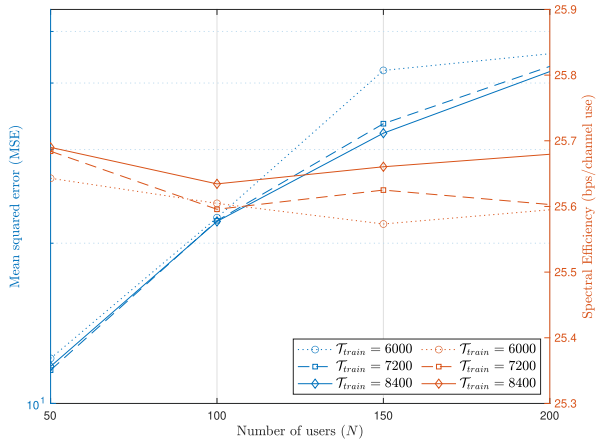


FIGURE 7. Analyzing the effect of the number of training data samples $\mathcal{T}_{train}$ on MSE and R_{sum} considering different numbers of users, N. ANN-NOMA-MEC with $b = 2$.

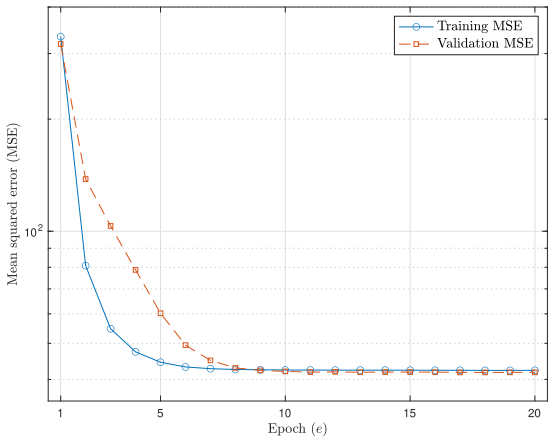


FIGURE 8. Evolution of ANN-NOMA-MEC learning through 20 epochs with $N = 200$ and $b = 2$. Training in offline mode.

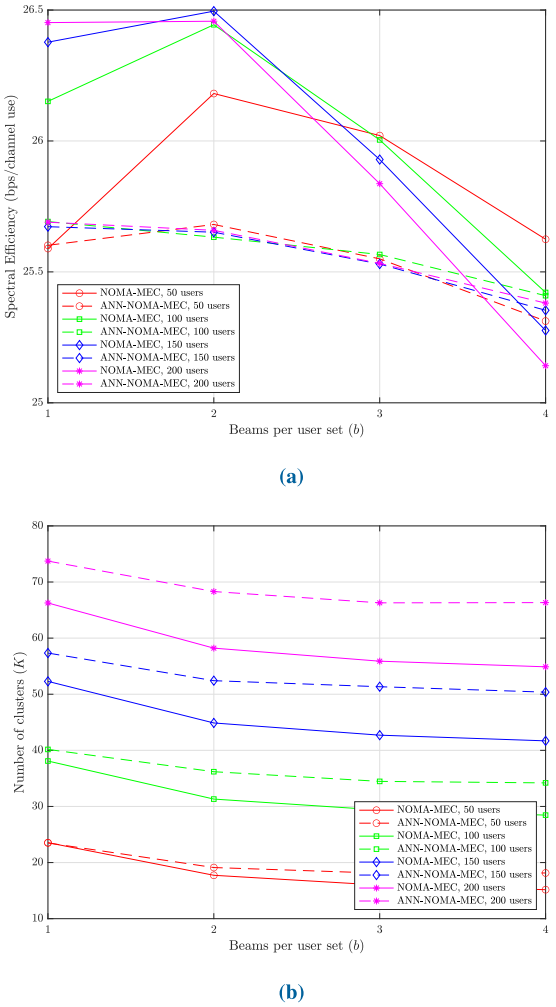
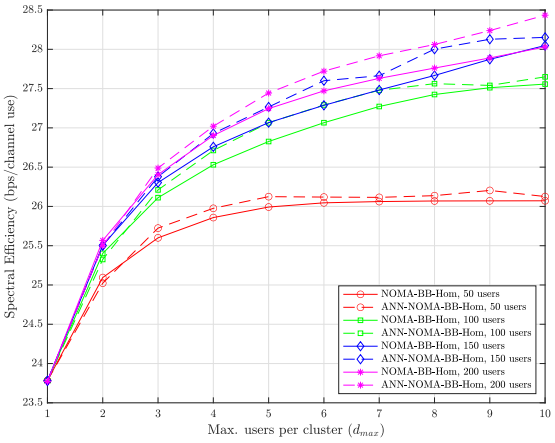


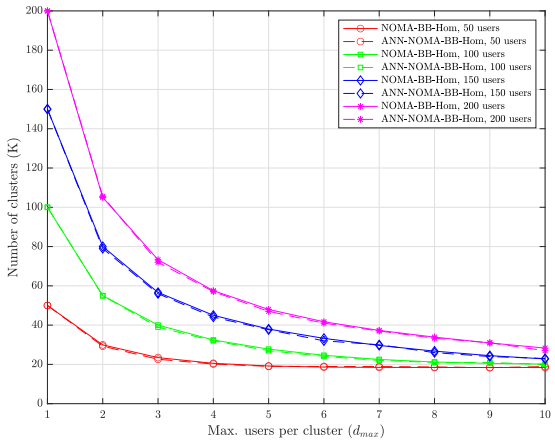
FIGURE 9. Evaluation of different ANN-NOMA-MEC models trained offline with different configurations of N and b, benchmarked with the NOMA-MEC heuristic. (a) Performance R_{sum} and (b) Number of user clusters, K.

a live network. In Fig. 11, we illustrate the evolution of both MSE training and MSE validation through 20 epochs for an ANN-NOMA-MEC network trained in offline mode focused on a system of $N = 200$ users and considering $b = 2$ best beams. As the figure shows, there is an aggressive reduction of the training MSE through the first 7 epochs where the ANN progressively adjusts its clustering formation estimation to the desired solutions. In parallel, a similar behavior can be observed with the MSE validation also, but with a less aggressive reduction rate. From epoch $e = 10$, both training MSE and validation MSE converge to an approximately constant performance measure that is kept until training ends. This confirms the hypothesis that running the ANNs offline up to epoch-10 allows the ANN-NOMA-MEC network to effectively learn to predict the clustering formations of the heterogeneous system at hand and, therefore, this ANN is suitable to be applied directly in live networks.

Finally, using the neural network parameters as described earlier, we run the ANN-NOMA-MEC for different settings of number of users, N and choices of number of best beams per user set, b, and compare it against the clustering outputs provided by the NOMA-MEC heuristic. Later, we do the same for ANN-NOMA-BB and compare it against NOMA-BB for homogeneous systems. In Fig. 9a and 9b, we show the effective sum rate R_{sum} and the number of clusters K, respectively, obtained from the estimates of clustering formations returned by the ANN-NOMA-MEC testing algorithm, considering $\mathcal{T}_{test} = 1800$ testing data samples. We plot that against the performance achieved by the clustering results from the baseline heuristic, NOMA-MEC. In general, based on Fig. 9a, we can see that the performance R_{sum} is quite stable between the different configurations of N and b, which suggests us that the networks have learned similar patterns from the NOMA-MEC heuristic. Although in the cases where $b = 2$ and $b = 3$ the spectral efficiency of the networks is visibly lower with respect to the original heuristic, the plot scale reflects that these values are very close. Therefore, we can infer that our ANN-NOMA-MEC models are able to attain the near-optimal R_{sum} performance as compared to



(a)



(b)

FIGURE 10. Evaluation of different ANN-NOMA-BB models trained offline with different number of users, N, benchmarked with the NOMA-BB heuristic for different values of the SIC decoding capability, $d_{\max}$. (a) Performance R_{sum} and (b) Number of user clusters, K.

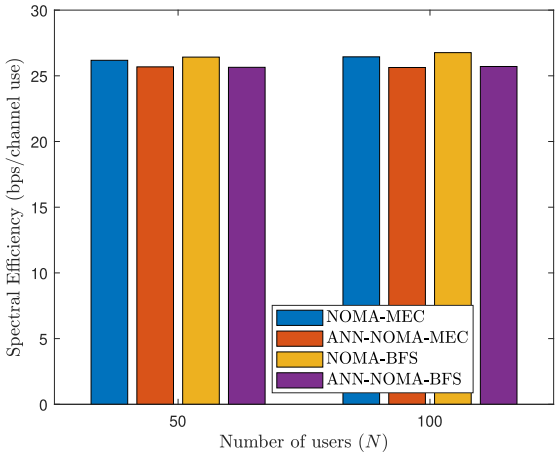


FIGURE 11. Comparison of a brute force search (BFS) with NOMA-MEC and ANNs trained on both the NOMA-MEC heuristic and the BFS.

NOMA-MEC heuristic. Furthermore, as seen in Fig. 9b, the original behavior of the NOMA-MEC heuristic is such that for each number of users in the heterogeneous system, the number of clusters required to serve the N users, i.e., K, decreases as b increases. This same behavior is followed by the ANN-NOMA-MEC models but showing slightly larger values. In any case, through Figs. 9a and 9b, we can assert that the ANN-NOMA-MEC scheme we have proposed is shown to be effective enough to closely follow the behavior of the NOMA-MEC heuristic. Similarly, in Fig. 10, we evaluate the homogeneous case by comparing the performance of ANN-NOMA-BB with its respective baseline heuristic, NOMA-BB. Unlike the heterogeneous case, the NOMA-BB heuristic is designed to always work with $b = 1$ while solving homogeneous systems where the SIC decoding capability $d_{\max}$ is an input parameter. This allows us to evaluate ANN-NOMA-BB models through different configurations of $d_{\max}$ as reflected in Fig. 10. Once again, we see that the performance of ANN-NOMA-BB closely follows the NOMA-BB heuristic it learned from both in terms of R_{sum} and number of clusters predicted, K. Since NOMA-BB operates on a homogeneous system which is simpler in terms of the underlying patterns of the data, ANN-NOMA-BB is able to more closely match or even slightly exceed the R_{sum} that NOMA-BB achieves; unlike ANN-NOMA-MEC which has to learn much more complex data patterns in heterogeneous systems and so we saw a slight decrease in R_{sum} compared to its baseline heuristic of NOMA-MEC.

The results from Fig. 9 and Fig. 10 show that the neural networks that were trained offline for both homogeneous and the more complex heterogeneous systems can achieve performance on par with the baseline heuristics it was trained on when applied in live networks. While the training and fine tuning of the neural network parameters is a tedious process, this can be done offline and off the BS compute resources, e.g., on cloud resources. When the trained ANNs are applied directly for NOMA clustering in live networks by the BS compute resources, it is a low complexity algorithm. This compared to the baseline heuristics of NOMA-MEC and NOMA-BB that execute thousands of operations to find the optimal cluster assignment on a millisecond granularity. Hence, the proposed ANN-NOMA-MEC and ANN-NOMA-BB allow for practical realizations of NOMA in mmWave systems that need to consider a mix of users with different SIC decoding capability constraints.

Finally, we show that the proposed approach generalizes well to datasets other than one prepared using the NOMA-MEC heuristic. To do this, we create a dataset for a heterogeneous system with $b = 2$ using a brute force search (BFS) of all possible clustering options as opposed to one obtained using the NOMA-MEC heuristic. We then train the same ANN on this dataset and term this as ANN-NOMA-BFS to indicate that it has been trained on the BFS dataset. The results are shown in Fig. 11 where it is firstly worth noting that BFS performs marginally better than the NOMA-MEC heuristic, as expected. The results in Fig. 11 also show that

the ANN-NOMA-MEC performs nearly on par as the ANN-NOMA-BFS, indicating there was not much performance lost from using the dataset generated using the NOMA-MEC heuristic as opposed to a full brute force search.

V. CONCLUSION

In this paper, we proposed a neural network aided machine learning approach to the user clustering and ordering problem in mmWave-NOMA systems that can factor in the individual SIC decoding capability of each user in the system. The proposed neural networks, called ANN-NOMA-MEC and ANN-NOMA-BB, are trained offline on datasets generated from running simulated settings using the NOMA-MEC and NOMA-BB heuristics. By training the neural network offline using cloud compute resources, the heavy computational steps are executed away from the BS compute resources. Instead, in the live network, where clustering decisions have to be made at a millisecond granularity, the trained neural network can be directly applied to output a clustering result when provided each user's mmWave channel and SIC decoding capability as input. Simulation results show the effectiveness of the ANN-NOMA-MEC and ANN-NOMA-BB schemes as the neural network trained on offline simulation data performs comparably with the NOMA-MEC and NOMA-BB heuristics that is applying computationally intensive algorithms to make every clustering decision in a live network.

This research can be extended to consider user clustering problems in digital beamforming systems where multiple clusters can be formed and served at the same time, leading to inter-cluster interference that needs to be mitigated. With a large and diverse enough dataset, it would be interesting to learn if artificial neural networks can also learn the underlying patterns involved in cancelling out the inter-cluster interference on top of the SIC decoding capability constraints as considered in this paper. Since ANN-NOMA-MEC relies on instantaneous channel conditions, another future direction is to consider the impacts of imperfect CSI on the performance. Thus, one extension of this work is to consider non-CSI based feedback, e.g., location based feedback, in the training of the neural network for making clustering decisions.

REFERENCES

[1] A. S. Rajasekaran, O. Maraqa, H. U. Sokun, H. Yanikomeroglu, and S. Al-Ahmadi, "User clustering in mmWave-NOMA systems with user decoding capability constraints for B5G networks," *IEEE Access*, vol. 8, pp. 209949–209963, 2020.

[2] J. Cui, Z. Ding, P. Fan, and N. Al-Dhahir, "Unsupervised machine learning-based user clustering in millimeter-wave-NOMA systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7425–7440, Nov. 2018.

[3] S. Norouzi, B. Champagne, and Y. Cai, "Joint optimization framework for user clustering, downlink beamforming, and power allocation in MIMO NOMA systems," *IEEE Trans. Commun.*, vol. 71, no. 1, pp. 214–228, Jan. 2023.

[4] O. Maraqa, A. S. Rajasekaran, S. Al-Ahmadi, H. Yanikomeroglu, and S. M. Sait, "A survey of rate-optimal power domain NOMA with enabling technologies of future wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2192–2235, 4th Quart., 2020.

[5] S. M. R. Islam, N. Avazov, O. A. Dobre, and K.-S. Kwak, "Power-domain non-orthogonal multiple access (NOMA) in 5G systems: Potentials and challenges," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 721–742, 2nd Quart., 2017.

[6] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, Jun. 2020.

[7] X. Chen, F.-K. Gong, G. Li, H. Zhang, and P. Song, "User pairing and pair scheduling in massive MIMO-NOMA systems," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 788–791, Apr. 2018.

[8] D. Marasinghe, N. Jayaweera, N. Rajatheva, and M. Latva-Aho, "Hierarchical user clustering for mmWave-NOMA systems," in *Proc. 2nd 6G Wireless Summit (6G SUMMIT)*, Mar. 2020, pp. 1–5.

[9] J. Ren, Z. Wang, M. Xu, F. Fang, and Z. Ding, "An EM-based user clustering method in non-orthogonal multiple access," *IEEE Trans. Commun.*, vol. 67, no. 12, pp. 8422–8434, Dec. 2019.

[10] M. Shahjalal, M. F. Ahmed, M. M. Alam, M. H. Rahman, and Y. M. Jang, "Fuzzy C-means clustering-based mMIMO-NOMA downlink communication for 6G ultra-massive interconnectivity," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Apr. 2021, pp. 421–424.

[11] Q. N. Le, V.-D. Nguyen, O. A. Dobre, N.-P. Nguyen, R. Zhao, and S. Chatzinotas, "Learning-assisted user clustering in cell-free massive MIMO-NOMA networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12872–12887, Dec. 2021.

[12] S. P. Kumaresan, T. C. Keong, L. C. Kwang, and N. Y. Hoe, "Adaptive user clustering for downlink nonorthogonal multiple access based 5G systems using brute-force search," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 11, p. e4098, Nov. 2020.

[13] S. P. Kumaresan, C. K. Tan, and Y. H. Ng, "Extreme learning machine (ELM) for fast user clustering in downlink non-orthogonal multiple access (NOMA) 5G networks," *IEEE Access*, vol. 9, pp. 130884–130894, 2021.

[14] S. P. Kumaresan, C. K. Tan, and Y. H. Ng, "Efficient user clustering using a low-complexity artificial neural network (ANN) for 5G NOMA systems," *IEEE Access*, vol. 8, pp. 179307–179316, 2020.

[15] M. S. Ali, H. Tabassum, and E. Hossain, "Dynamic user clustering and power allocation for uplink and downlink non-orthogonal multiple access (NOMA) systems," *IEEE Access*, vol. 4, pp. 6325–6343, 2016.

[16] K. S. Ali, M. Haenggi, H. El Sawy, A. Chaaban, and M.-S. Alouini, "Downlink non-orthogonal multiple access (NOMA) in Poisson networks," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1613–1628, Feb. 2019.

[17] Z. Ding, R. Schober, and H. V. Poor, "Unveiling the importance of SIC in NOMA systems—Part 1: State of the art and recent findings," *IEEE Commun. Lett.*, vol. 24, no. 11, pp. 2373–2377, Nov. 2020.

[18] Z. Ding, P. Fan, and H. V. Poor, "Random beamforming in millimeter-wave NOMA networks," *IEEE Access*, vol. 5, pp. 7667–7681, 2017.

[19] J. Cui, Y. Liu, Z. Ding, P. Fan, and A. Nallanathan, "Optimal user scheduling and power allocation for millimeter wave NOMA systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1502–1517, Mar. 2018.

[20] K. Wang, J. Cui, Z. Ding, and P. Fan, "Stackelberg game for user clustering and power allocation in millimeter wave-NOMA systems," *IEEE Trans. Wireless Commun.*, vol. 18, no. 5, pp. 2842–2857, May 2019.

[21] M. Shahjalal, M. H. Rahman, M. O. Ali, B. Chung, and Y. M. Jang, "User clustering techniques for massive MIMO-NOMA enabled mmWave/THz communications in 6G," in *Proc. 12th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Aug. 2021, pp. 379–383.

[22] S. Kiani, M. Dong, S. ShahbazPanahi, G. Boudreau, and M. Bavand, "Learning-based user clustering in NOMA-aided MIMO networks with spatially correlated channels," *IEEE Trans. Commun.*, vol. 70, no. 7, pp. 4807–4821, Jul. 2022.

[23] C. B. Issaid, C. Anton-Haro, X. Mestre, and M.-S. Alouini, "User clustering for MIMO NOMA via classifier chains and gradient-boosting decision trees," *IEEE Access*, vol. 8, pp. 211411–211421, 2020.

[24] S. Solaiman, L. Nassef, and E. Fadel, "User clustering and optimized power allocation for D2D communications at mmWave underlaying MIMO-NOMA cellular networks," *IEEE Access*, vol. 9, pp. 57726–57742, 2021.

[25] I. Orikumhi, C. Y. Leow, and S. Kim, "Location-aided user selection and sum-rate analysis for mmWave NOMA," *J. Commun. Netw.*, vol. 25, no. 1, pp. 1–15, 2023.

[26] A. S. Rajasekaran, H. U. Sokun, O. Maraqa, H. Yanikomeroglu, and S. Al-Ahmadi, "Vision-assisted user clustering for robust mmWave-NOMA systems," 2022, *arXiv:2205.14716*.
[27] N. S. Mouni, M. P. Reddy, A. Kumar, and P. K. Upadhyay, "Enhanced user pairing and power allocation strategies for downlink NOMA systems with imperfections in SIC," in *Proc. 15th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2023, pp. 457–461.
[28] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artif. Intell. Rev.*, vol. 53, pp. 5929–5955, May 2020.

ADITYA S. RAJASEKARAN (Member, IEEE) received the B.Eng. (Hons.) and M.Eng. degrees in systems and computer engineering from Carleton University, Ottawa, ON, Canada, in 2014 and 2017, respectively, where he is currently pursuing the Ph.D. degree in systems and computer engineering.

He is with Ericsson Canada Inc., where he has been a Software Developer, since 2014. He has also been involved in the physical layer development work with Ericsson's 5G new radio (NR) solutions. His research interests include wireless technology solutions aimed toward 5G and beyond cellular networks, including non-orthogonal multiple access solutions.

HALIM YANIKOMEROGLU (Fellow, IEEE) received the B.Sc. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 1990, and the M.A.Sc. degree in electrical engineering (now ECE) and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Canada, in 1992 and 1998, respectively.

Since 1998, he has been with the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, where he is currently a Full Professor. He has given more than 110 invited seminars, keynotes, panel talks, and tutorials in the last five years. He has supervised or hosted over 150 postgraduate researchers in his laboratory at Carleton University. His extensive collaborative research with industry resulted in 39 granted patents. His research interests include wireless communications and networks, with a special emphasis on non-terrestrial networks (NTN) in recent years.

Dr. Yanikomeroglu is a fellow of the Engineering Institute of Canada (EIC) and the Canadian Academy of Engineering (CAE). He is also a member of the IEEE ComSoc Governance Council, IEEE ComSoc GIMS, IEEE ComSoc Conference Council, and IEEE PIMRC Steering Committee. He received several awards for his research, teaching, and service, including the IEEE ComSoc Fred W. Ellersick Prize, in 2021, the IEEE VTS Stuart Meyer Memorial Award, in 2020, and the IEEE ComSoc Wireless Communications TC Recognition Award, in 2018. He received the Best Paper Award from the IEEE Competition on Non-Terrestrial Networks for B5G and 6G, in 2022 (grand prize), IEEE ICC 2021, and IEEE WISEE 2021 and 2022. He served as the general chair and the technical program chair for several IEEE conferences. He is also serving as the Chair for the Steering Committee of IEEE's Flagship Wireless Event and the Wireless Communications and Networking Conference (WCNC). He has also served on the editorial boards for various IEEE periodicals. He is a Distinguished Speaker of the IEEE Communications Society and the IEEE Vehicular Technology Society, and an Expert Panelist of the Council of Canadian Academies (CCA|CAC).