Throughput Optimization in Grant-Free NOMA with Deep Reinforcement Learning

Rui Huang*, Vincent W.S. Wong*, and Robert Schober†
*Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
†Friedrich-Alexander University of Erlangen-Nürnberg, Germany
email: {ruihuang, vincentw}@ece.ubc.ca, robert.schober@fau.de

***Abstract*—Grant-free non-orthogonal multiple access (GF-NOMA) is a promising paradigm for reducing the access delay and improving the spectrum efficiency. As the signals of multiple users are superimposed in GF-NOMA systems, each user is required to select a user-specific pilot sequence to distinguish its own signal from the signals of other users. Packet collisions in the uplink occur when multiple users select the same pilot sequence. In this paper, we first formulate a pilot sequence selection problem for aggregate throughput maximization in GF-NOMA systems. We then design a deep reinforcement learning (DRL)-based distributed algorithm for each user to select its pilot sequence via learning from the past pilot sequence selections. The proposed algorithm does not rely on information exchange between the users and does not require centralized scheduling by the base station. Packet-level simulations show that, for the considered system parameters, the proposed DRL distributed algorithm can achieve an average aggregate throughput which is within 90% of the optimal value, and has a better performance than both acknowledgement-based and random selection GF-NOMA schemes.**

I. Introduction

Grant-free non-orthogonal multiple access (GF-NOMA) is a promising paradigm for supporting massive machine-type communication (mMTC) use cases and the Internet of Things (IoT) in fifth generation (5G) wireless networks as it enhances the spectrum efficiency and reduces the access delay [1], [2]. For uplink non-orthogonal multiple access (NOMA), multiple users can share the same resource to transmit their packets simultaneously to the base station. In GF-NOMA, a user transmits its packets to the base station without sending an access request to the base station beforehand. This reduces the signaling overhead required for coordination between the base station and the users.

To fully exploit the benefits of GF-NOMA, two challenges have to be overcome. First, in GF-NOMA systems, signals from multiple users are superimposed. Each user should choose a user-specific pilot sequence that distinguishes its signal from the signals of other users to ensure successful channel estimation and decoding at the receiver [3]. Due to the lack of centralized scheduling, packet collisions occur when multiple users select the same pilot sequence, which can lead to decoding failure and throughput degradation. Second, since it is difficult for a user to exchange information with other users in GF-NOMA systems, the pilot sequence selection should be performed by the users in a distributed manner.

The authors in [4] proposed an acknowledgement (ACK)-based scheduling scheme, where a user who experienced a packet collision will select a new pilot sequence from the rest of the pilot sequences which have not yet been selected by other users and retransmit the packet. The authors in [5] designed an ACK frame to provide feedback for grant-free uplink transmission. Through the broadcast of an ACK, the base station allocates an exclusive pilot sequence to users that have suffered a packet collision. For ACK-based solutions [4], [5], although a collision is resolved in the retransmission phase, collisions can still occur when a user transmits a packet for the first time as the scheduling is performed only after a collision occurs. The authors in [6] proposed a pilot sequence allocation scheme, in which the base station allocates exclusive pilot sequences to users that have higher probabilities of transmitting packets in the next time slot. However, the allocation scheme in [6] requires centralized scheduling and accurate estimation of the transmit probability.

Deep reinforcement learning (DRL) is a learning technique based on deep neural networks (DNNs). It does not rely on a pre-established tractable system model and does not require information exchange between users [7], [8]. DRL is a promising technique for the design of distributed algorithms to improve pilot sequence selection and avoid packet collision. Nevertheless, to the best of our knowledge, the application of DRL for the design of GF-NOMA protocols has not been considered yet, while some insights can be obtained from the application of DRL for spectrum access [9], [10]. The authors in [9] used DRL to design a distributed multi-channel access algorithm. Furthermore, DRL was employed to study cooperative and non-cooperative distributed channel access of multiple users in [10]. Unlike the distributed channel access problem, where a channel is occupied by only one user, for the considered pilot sequence selection problem, users cannot avoid collisions through sensing the channel in advance as multiple users share the same physical layer resource.

To address the aforementioned issues, in this paper, we propose a DRL-based distributed pilot sequence selection algorithm for aggregate throughput maximization in GF-NOMA systems. Our contributions are as follows:

- We formulate the pilot sequence selection problem for throughput optimization in GF-NOMA systems as a combinatorial optimization problem.
- We design a distributed DRL framework for GF-NOMA

systems, where a multi-layer DNN module is used to aggregate experience throughout the access process. We design an offline training algorithm that exploits the *system transition history* to efficiently train the DNN module.
- We propose a distributed online algorithm for users to select their pilot sequences exploiting the pre-trained DNN module, without relying on centralized scheduling or information exchange between users.
- We conduct packet-level simulations to evaluate the performance of the proposed DRL algorithm. Results show that, for the considered system, the proposed algorithm can achieve an aggregate throughput which is within 90% of the optimum. The proposed algorithm can maintain a higher aggregate throughput for different numbers of pilot sequences and users than two other previously proposed GF-NOMA schemes.

This paper is organized as follows. The system model and problem formulation are introduced in Section II. The DRL-based GF-NOMA pilot sequence selection algorithm is presented in Section III. Simulation results are given in Section IV. Conclusions are drawn in Section V.

II. System Model and Problem Formulation

We consider a GF-NOMA system with one base station serving multiple users. The base station and each user are equipped with one antenna. Time is slotted into intervals of equal duration. The time interval $[t, t+1)$ is referred to as time slot t, where $t \in \mathcal{T} = \{0, 1, 2, \ldots, T-1\}$. In each time slot, the base station assigns one physical layer time-frequency resource block, referred to as the basic physical resource (BPR), for GF-NOMA transmission. We assume the base station assigns K pilot sequences to the BPR in each time slot, and $\mathcal{K} = \{1, 2, \ldots, K\}$ is the set of pilot sequence indices. Welch bound equality sequences, Grassmannian sequences, or other types of sparse spreading sequences can be used as pilot sequences [1, Section 5.1]. In each time slot, N users share one BPR for transmitting their packets using code-domain NOMA. The set of users is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$.

The timing sequence diagram for GF-NOMA is illustrated in Fig. 1. The base station first informs the users about the BPR and the K available pilot sequences via radio resource control (RRC) signaling [1]. When a user decides to transmit, it selects one of the K available pilot sequences. Using the selected pilot sequence, the user then sends its packet to the base station. We define binary variable $g_{nk}(t) \in \{0, 1\}$, where $g_{nk}(t)$ is equal to 1 if user n selects the k-th pilot sequence, $k \in \mathcal{K}$, in time slot $t \in \mathcal{T}$. Otherwise, $g_{nk}(t)$ is equal to 0. Since a user can select at most one pilot sequence in each time slot, we have

$$\sum_{k \in \mathcal{K}} g_{nk}(t) \leq 1, \quad n \in \mathcal{N},\, t \in \mathcal{T}. \tag{1}$$

User n does not transmit in time slot t if $\sum_{k \in \mathcal{K}} g_{nk}(t) = 0$. We further define vector $\boldsymbol{g}_n(t) = (g_{n1}(t), g_{n2}(t), \ldots, g_{nK}(t))$ as the pilot sequence selection of user n in time slot t.

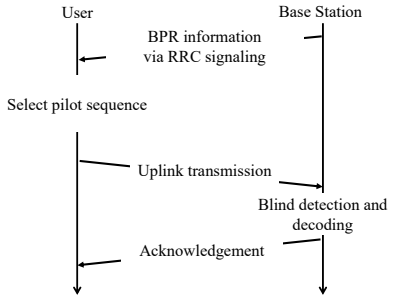

Fig. 1. The timing sequence diagram for GF-NOMA. The base station configures the BPR via RRC signaling. Each user then selects a pilot sequence and transmits its packet to the base station. After decoding, the base station sends an ACK to the users.

We define $n_k(t)$ to be the number of users that select the k-th pilot sequence in time slot t. We have

$$n_k(t) = \sum_{n \in \mathcal{N}} g_{nk}(t), \quad k \in \mathcal{K},\, t \in \mathcal{T}. \tag{2}$$

Furthermore, $\mathcal{S}(t)$ denotes the set of users who select a pilot sequence that is not chosen by other users in time slot t. That is,

$$\mathcal{S}(t) = \left\{ n \,\middle|\, \sum_{k \in \mathcal{K}} \mathbb{1}(n_k(t) = 1) g_{nk}(t) \;=\; 1,\; n \in \mathcal{N} \right\}, t \in \mathcal{T},$$

where $\mathbb{1}(\cdot)$ is the indicator function.

We assume the uplink channel suffers from Rayleigh fading and path loss. We assume perfect channel estimation. We denote $d_n(t)$ and $h_n(t)$ as the distance and channel gain between user n and the base station in time slot t, respectively. We denote $P_n^{\text{tx}}(t) \in [0,\, P^{\max}]$ as the transmit power of user n in time slot t, where $P^{\max}$ is the maximum transmit power. We define $P_n^{\text{rx}}(t)$ as the received power of user n at the base station in time slot t. We have

$$P_n^{\text{rx}}(t) = G_n(t) P_n^{\text{tx}}(t), \quad n \in \mathcal{N},\, t \in \mathcal{T}, \tag{3}$$

where $G_n(t) = |h_n(t)|^2 d_n^{-\beta}(t)$ and β is the path loss exponent. We adopt open loop power control [1, Section 9.2], such that the received signal powers at the base station are equal to a constant P_0 for all users.

To decode the signals received in the considered BPR, the base station employs blind detection to detect the pilot sequences. Due to potential pilot sequence collisions, we assume the base station can only successfully decode the data of the users in set $\mathcal{S}(t)$. For user $n \in \mathcal{N}$, we define $r_n(t)$ to be a binary indicator that specifies whether its signal is successfully decoded in time slot t. We have

$$r_n(t) = \begin{cases} 1, & \text{if } n \in \mathcal{S}(t), \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

After decoding, the base station broadcasts an ACK to inform users about the decoding results. Our objective is to maximize the time average aggregate throughput in GF-NOMA systems, which leads to the following optimization problem

$$\begin{aligned} \underset{\boldsymbol{g}_n(t),\, n \in \mathcal{N},\, t \in \mathcal{T}}{\text{maximize}} \quad & \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{n \in \mathcal{N}} r_n(t) \\ \text{subject to} \quad & \text{constraints (1) and (4).} \end{aligned} \tag{5}$$

Problem (5) is a combinatorial optimization problem. Obtaining the optimal solution of (5) requires centralized computation. Besides, centralized scheduling is needed for the implementation of the optimal solution of (5) in GF-NOMA systems. In the next section, we propose a DRL-based online pilot sequence selection algorithm to obtain a suboptimal solution of (5) in a fully distributed manner.

III. DRL-based Pilot Sequence Selection

DRL is a powerful tool for finding distributed solutions for combinatorial optimization problems that require neither centralized scheduling nor information exchange between the users. In this section, we develop a multi-layer DNN module along with a training framework for efficiently solving problem (5). Moreover, we propose an online algorithm that allows the users to select their pilot sequences in a distributed manner.

A. Decision Process and Deep Q-learning

We model the pilot sequence selection of each user $n \in \mathcal{N}$ as a decision process $(\boldsymbol{s}(t), \boldsymbol{a}_n(t), R_n(t))$, where the details are as follows:

1) State $\boldsymbol{s}(t)$: We define variable $v_k(t) \in \{-1, 0, 1\}$, $k \in \mathcal{K}$, as the indicator for the decoding state of the k-th pilot sequence at the beginning of the current time slot $t \in \mathcal{T}$. $v_k(t)$ depends on the pilot sequence selection of the users in the previous time slot $t-1$. Specifically, $v_k(t)$ is equal to 1 if the signal using the k-th pilot sequence was decoded successfully in time slot $t-1$. We set $v_k(t)$ to be -1 if the signal using the k-th pilot sequence was not decoded successfully in time slot $t-1$. $v_k(t)$ is equal to 0 if no signal using the k-th pilot was transmitted in time slot $t-1$.

In time slot t, the state $\boldsymbol{s}(t)$ consists of the decoding states of all pilot sequences. We have

$$\boldsymbol{s}(t) = (v_1(t), v_2(t), \ldots, v_K(t)), \quad t \in \mathcal{T}. \tag{6}$$

2) Action $\boldsymbol{a}_n(t)$: In time slot t, the action profile of user n consists of its own pilot sequence selection $\boldsymbol{g}_n(t)$. We have

$$\boldsymbol{a}_n(t) = \boldsymbol{g}_n(t), \quad n \in \mathcal{N},\ t \in \mathcal{T}. \tag{7}$$

We define $\mathcal{A}_n \subseteq \{0,1\}^K$ as the set of feasible $\boldsymbol{g}_n(t)$ of user $n \in \mathcal{N}$ which satisfy constraint (1).

3) Reward $R_n(t)$: At the end of time slot t, the users determine their rewards based on the ACK received from the base station. For user $n \in \mathcal{N}$, the reward $R_n(t)$ is given by

$$R_n(t) = r_n(t), \quad n \in \mathcal{N},\ t \in \mathcal{T}, \tag{8}$$

where $r_n(t)$ is specified in (4).

To maximize its reward during the decision process, user n *may* apply traditional Q-learning [11], where the expected reward of selecting $\boldsymbol{a}_n(t)$ under state $\boldsymbol{s}(t)$ in time slot t is given by the *Q-value* $Q(\boldsymbol{s}(t), \boldsymbol{a}_n(t))$. To maximize its reward, user n determines its action from

$$\boldsymbol{a}_n(t) = \arg\max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q(\boldsymbol{s}(t), \boldsymbol{a}_n). \tag{9}$$

The Q-value is updated during the decision process. Given the state $\boldsymbol{s}(t)$ in time slot t, if user n chooses action $\boldsymbol{a}_n(t)$, receives reward $R_n(t)$, and observes the next state $\boldsymbol{s}_n(t+1)$, then it will update the Q-value as

$$Q(\boldsymbol{s}(t), \boldsymbol{a}_n(t)) \leftarrow R_n(t) + \gamma \max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q(\boldsymbol{s}_n(t+1), \boldsymbol{a}_n),$$

where γ is a constant discount factor. To obtain $\boldsymbol{a}_n(t)$ based on (9), in traditional Q-learning, each user has to maintain a Q-table to store the Q-values of all possible actions under given state $\boldsymbol{s}(t)$. However, the size of the Q-table increases with the cardinality of the action and state spaces, which makes Q-learning costly in terms of computation and memory.

Deep Q-learning [7] has been proposed to tackle the aforementioned issues. In deep Q-learning, the Q-value is approximated by DNNs through the establishment of a mapping between each state and the corresponding Q-values of all actions. Specifically, user n uses a DNN with total Z learnable parameters (i.e., the weights and biases) to approximate the Q-values. We define $\mathcal{Z} = \{1, 2, \ldots, Z\}$ to be the set of the indices of all parameters. We denote the parameters of the DNN of user n as a vector $\boldsymbol{\Phi}_n = (\phi_n^{(1)}, \phi_n^{(2)}, \ldots, \phi_n^{(Z)})$, where $\phi_n^{(z)}$ is the z-th parameter for $z \in \mathcal{Z}$. We initialize all parameters in $\boldsymbol{\Phi}_n$ to values drawn randomly from a uniform distribution. Parameters $\boldsymbol{\Phi}_n$ are then updated during the training process to improve the accuracy of the Q-value approximation based on the system transition history. In the current time slot t, the system transition history of user n consists of the *system transition tuples* $(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau), \boldsymbol{s}_n(\tau+1), R_n(\tau))$ with *time index* $\tau \in \{0, 1, \ldots, t-1\}$, which describes the system transition from time slot τ to time slot $\tau+1$. We denote the Q-value for $\boldsymbol{s}(t)$ and $\boldsymbol{a}_n(t)$, which is approximated by a DNN employing parameters $\boldsymbol{\Phi}_n$, as $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}(t), \boldsymbol{a}_n(t))$.

To update parameters $\boldsymbol{\Phi}_n$ based on the system transition tuple, in each training iteration, we first save the current parameters before update as $\widehat{\boldsymbol{\Phi}}_n$ and use $\widehat{\boldsymbol{\Phi}}_n$ to obtain the target of the Q-value approximation, which is $R_n(\tau) + \gamma \max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q_{\widehat{\boldsymbol{\Phi}}_n}(\boldsymbol{s}_n(\tau+1), \boldsymbol{a}_n)$. We then find the $\boldsymbol{\Phi}_n$ that minimizes the difference between the target and the approximated Q-value [7]. That is,

$$\begin{aligned} \arg\min_{\boldsymbol{\Phi}_n} \quad & \frac{1}{2}\big(R_n(\tau) + \gamma \max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q_{\widehat{\boldsymbol{\Phi}}_n}(\boldsymbol{s}_n(\tau+1), \boldsymbol{a}_n) \\ & - Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))\big)^2, \end{aligned} \tag{10}$$

where $\tau \in \{0, 1, \ldots, t-1\}, n \in \mathcal{N}$. To solve problem (10), we apply a stochastic gradient decent (SGD) algorithm to update parameters $\boldsymbol{\Phi}_n$. Specifically, parameters $\boldsymbol{\Phi}_n$ of the DNN of user n are updated as [12]

$$\begin{aligned} \boldsymbol{\Phi}_n \leftarrow\ & \boldsymbol{\Phi}_n - \alpha\, \nabla Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))\, \big(R_n(\tau) \\ & + \gamma \max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q_{\widehat{\boldsymbol{\Phi}}_n}(\boldsymbol{s}_n(\tau+1), \boldsymbol{a}_n) \\ & - Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))\big), \end{aligned} \tag{11}$$

where α is a positive learning rate.

We use backpropagation algorithm [13, Ch. 6] to determine the gradient $\nabla Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))$ in (11). In particular, we

first determine the partial derivatives of the approximated Q-value $Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))$ with respect to the parameters of the DNN layer that directly yields the Q-value. Then, the partial derivatives of $Q_{\mathbf{\Phi}_n}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))$ with respect to the remaining parameters in $\mathbf{\Phi}_n$ can be determined by recursively applying the chain rule based on the previously obtained partial derivatives.

After updating parameters $\mathbf{\Phi}_n$ with a sufficient number of system transition tuples, the DNN can accurately approximate the Q-values. Based on the pre-trained DNN with updated parameters $\mathbf{\Phi}_n$, user n can determine its action as

$$\boldsymbol{a}_n(t) = \underset{\boldsymbol{a}_n \in \mathcal{A}_n}{\arg\max}\, Q_{\mathbf{\Phi}_n}(\boldsymbol{s}(t), \boldsymbol{a}_n). \tag{12}$$

To obtain $\boldsymbol{a}_n(t)$ based on (12), we feed $\boldsymbol{s}(t)$ into the DNN and determine $\boldsymbol{a}_n(t)$ based on the output of the DNN. Instead of maintaining a large Q-table, with deep Q-learning, user n only needs to store parameters $\mathbf{\Phi}_n$.

B. Proposed DNN Architecture

To determine the pilot sequence selection that maximizes the reward in (8) based on deep Q-learning, we design a corresponding multi-layer DNN module. The proposed structure of the multi-layer DNN module is illustrated in Fig. 2. The DNN layers and their functionalities are as follows:

1) Input Layer: The input layer collects the state and feeds it to the DNN. For the multi-layer DNN module of user n, the input is the state $\boldsymbol{s}(t)$.

2) Long Short-Term Memory (LSTM) Layer: LSTM is a deep recurrent neural network that can aggregate experience more efficiently than convolutional neural networks for time series learning problems [8]. Therefore, we use an LSTM layer with H hidden units to learn from the system transition history and approximate the Q-values.

3) Output Layer: The LSTM layer is connected to the output layer to generate the Q-values. Specifically, we use two fully connected (FC) layers with one rectified linear unit (ReLU) layer to map the output of the LSTM layer to the Q-values. The output is a vector of size $K+1$. For $k \in \mathcal{K}$, the $(k+1)$-th entry is the Q-value for selecting the k-th pilot sequence in time slot t for state $\boldsymbol{s}(t)$, while the first entry is the Q-value for not transmitting in time slot t.

In the aforementioned network structure, the number of parameters in the LSTM layer is $4H(K+H)$. The size of the first FC layer in the output layer is set to $H \times 64$, where H and 64 are the dimensions of the input and output, respectively. The first FC layer then contains $64H$ weights and 64 biases. The second FC layer in the output layer employs a size of $64 \times (K+1)$, with $64(K+1)$ weights and $K+1$ biases. The input layer and ReLU layer do not have learnable parameters. Therefore, the total number of parameters Z in the proposed network structure is

$$Z = \underbrace{4H(K+H)}_{\text{LSTM layer}} + \underbrace{64H+64}_{\text{first FC layer}} + \underbrace{64(K+1)+(K+1)}_{\text{second FC layer}}.$$

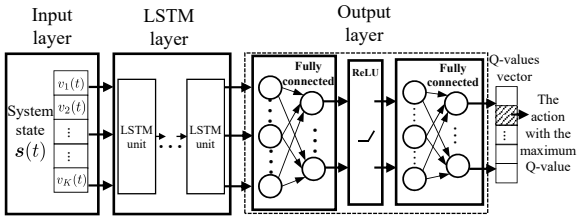

Fig. 2. The proposed network structure consists of an input layer, an LSTM layer, and an output layer.

C. Training Framework

The multi-layer DNN module has to be trained for accurate Q-value approximation. We propose a DRL training framework in which users generate and accumulate the system transition history and forward it to the base station along with their data. The base station uses the system transition history to train the DNN module and sends the DNN parameters back to the users. In the proposed framework, users do not have to perform extensive computations during the training process, and meanwhile can leverage the pre-trained DNNs to improve their decisions.

In order to store the system transition history for training the DNNs, the base station maintains a long-term *replay* for user $n \in \mathcal{N}$, which consists of a large number of system transition tuples. We denote the set of time indices of the system transition tuples stored in the long-term replay of user n by $\mathcal{T}_n$. User n also maintains a local short-term replay, in which it stores the system transition tuple $(\boldsymbol{s}(t), \boldsymbol{a}_n(t), \boldsymbol{s}_n(t+1), R_n(t))$ at the end of time slot t. While the long-term replay at the base station records the system transition history in a long time scale, the local short-term replay only maintains the system transition tuples of the recent time slots. When user n transmits its packet, it uploads its local short-term replay to the base station. The base station then updates the long-term replay with the short-term replay from user n.

Recall that when updating the parameters with (10), the parameters for obtaining the target Q-value (i.e., $\widehat{\mathbf{\Phi}}_n$), and the parameters we update (i.e., $\mathbf{\Phi}_n$) come from the same DNN. As the parameters $\mathbf{\Phi}_n$ cannot accurately approximate the Q-value in the early stages of training, errors in the approximation of the Q-value will lead to an overestimation of the target Q-value and consequently affect the update of $\mathbf{\Phi}_n$ [14]. To tackle these issues, we use deep Q-learning with *double Q-learning* [14] to update the parameters, where we decompose the Q-value approximation and action selection by using one DNN module to estimate the Q-values and another DNN module to determine the pilot sequence selection. Furthermore, we refer to the DNN module for Q-value estimation as the *target DNN module*. We use $\mathbf{\Phi}_n^{\text{tar}}$ to denote the parameters of the target DNN module of user n. We refer to the DNN module for determining pilot sequence selection as the *policy DNN module*. We use $\mathbf{\Phi}_n^{\text{pol}}$ to denote the parameters of the policy DNN module of user n. Both the policy and target DNN modules employ the same network structure as introduced in the previous subsection.

The proposed training framework is illustrated in Fig. 3,

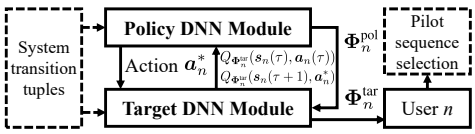

Fig. 3. Illustration of the training framework. The policy DNN module determines the action $\boldsymbol{a}_n^*$ and feeds it to the target DNN module. The target DNN module then estimates the Q-values and feeds them to the policy DNN module for updating $\boldsymbol{\Phi}_n^{\text{pol}}$. $\boldsymbol{\Phi}_n^{\text{tar}}$ is updated periodically by copying $\boldsymbol{\Phi}_n^{\text{pol}}$.

and the training algorithm is summarized in Algorithm 1. In Lines $3-5$ of Algorithm 1, the base station first determines the difference between the target Q-value and the approximated Q-value. In particular, the base station samples system transition tuples $(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau), \boldsymbol{s}_n(\tau+1), R_n(\tau)), \tau \in \mathcal{T}_n$ of $m = 20$ consecutive time slots [8], and updates $\boldsymbol{\Phi}_n^{\text{pol}}$ by [14]

$$\boldsymbol{a}_n^* := \arg\max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q_{\boldsymbol{\Phi}_n^{\text{pol}}}(\boldsymbol{s}_n(\tau+1)\boldsymbol{a}_n), \tag{13a}$$

$$\arg\min_{\boldsymbol{\Phi}_n^{\text{pol}}} \quad \frac{1}{2}(R_n(\tau) + \gamma\, Q_{\boldsymbol{\Phi}_n^{\text{tar}}}(\boldsymbol{s}_n(\tau+1), \boldsymbol{a}_n^*) - Q_{\boldsymbol{\Phi}_n^{\text{tar}}}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau)))^2. \tag{13b}$$

In Line 4, action $\boldsymbol{a}_n^*$ in (13a) is determined by feeding $\boldsymbol{s}_n(\tau+1)$ into the policy DNN module and selecting the action that corresponds to the entry with the maximum value in the output. In Line 5, the term $Q_{\boldsymbol{\Phi}_n^{\text{tar}}}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))$ in (13b) is obtained by feeding the state $\boldsymbol{s}_n(\tau)$ into the target DNN module and selecting the values of the entries of the output that correspond to $\boldsymbol{a}_n(\tau)$. The term $Q_{\boldsymbol{\Phi}_n^{\text{tar}}}(\boldsymbol{s}_n(\tau+1), \boldsymbol{a}_n^*)$ in (13b) is obtained by feeding the state $\boldsymbol{s}_n(\tau+1)$ into the target DNN module and selecting the values of the entries of the output that correspond to $\boldsymbol{a}_n^*$.

In Line 6, the base station updates the parameters $\boldsymbol{\Phi}_n^{\text{pol}}$ by using the SGD algorithm [12] to solve problem (13b). The parameters of the policy DNN module are updated in every iteration, while the parameters of the target DNN module are updated less frequently to avoid potential overestimation. In particular, in Line 7, we update the parameters $\boldsymbol{\Phi}_n^{\text{pol}}$ every M iterations by copying the parameters from the policy DNN module, i.e., $\boldsymbol{\Phi}_n^{\text{tar}} \leftarrow \boldsymbol{\Phi}_n^{\text{pol}}$. The base station will send the parameters of the target DNN module $\boldsymbol{\Phi}_n^{\text{tar}}$ to user n.

D. Proposed Online Algorithm

To determine the pilot sequence selection in the current time slot t, each user $n \in \mathcal{N}$ maintains a *local* DNN module with parameters $\boldsymbol{\Phi}_n$. User n receives the updated parameters of the target DNN module $\boldsymbol{\Phi}_n^{\text{tar}}$ from the base station. As the local DNN module has the same structure as the target DNN module, user n updates $\boldsymbol{\Phi}_n$ as $\boldsymbol{\Phi}_n \leftarrow \boldsymbol{\Phi}_n^{\text{tar}}$. User n then applies an ϵ-greedy policy to determine its pilot sequence selection. Specifically, the pilot sequence selection of user n in the current time slot t is determined by

$$\boldsymbol{a}_n(t) = \begin{cases} \arg\max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}(t), \boldsymbol{a}_n), & \text{with probability } 1-\epsilon \\ \text{random selection}, & \text{with probability } \epsilon, \end{cases} \tag{14}$$

Algorithm 1 Offline Training Algorithm for the Target and Policy DNN Modules in Each Training Iteration

1: **for** $n \in \mathcal{N}$ **do**
2: Update the replay if a new transition history is received.
3: Sample the system transition tuples $(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau), \boldsymbol{s}_n(\tau+1), R_n(\tau)), \tau \in \mathcal{T}_n$.
4: Determine $\boldsymbol{a}_n^*$ with the policy DNN module.
5: Determine $Q_{\boldsymbol{\Phi}_n^{\text{tar}}}(\boldsymbol{s}_n(\tau), \boldsymbol{a}_n(\tau))$ and $Q_{\boldsymbol{\Phi}_n^{\text{tar}}}(\boldsymbol{s}_n(\tau+1), \boldsymbol{a}_n^*)$ with the target DNN module.
6: Update $\boldsymbol{\Phi}_n^{\text{pol}}$ by using the SGD algorithm to solve (13b).
7: Update the parameters of the target DNN module $\boldsymbol{\Phi}_n^{\text{tar}}$ as $\boldsymbol{\Phi}_n^{\text{tar}} \leftarrow \boldsymbol{\Phi}_n^{\text{pol}}$ every M iterations.
8: **end for**

Algorithm 2 Online Pilot Sequence Selection Algorithm for User $n \in \mathcal{N}$ in Time Slot $t \in \mathcal{T}$

1: Determine the pilot sequence selection according to the ϵ-greedy policy in (14).
2: **if** $\sum_{k \in \mathcal{K}} g_{nk}(t) = 1$ **then**
3: Send packet along with local short-term replay to base station.
4: **else**
5: User n does not transmit in time slot t.
6: **end if**
7: Calculate $R_n(t)$ and observe $\boldsymbol{s}_n(t+1)$ based on the received ACK, and store the system transition tuple.

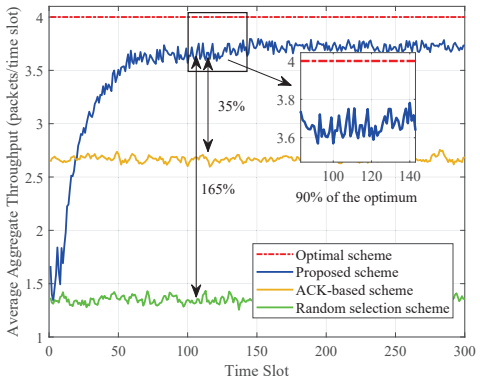

Fig. 4. Average aggregate throughput versus time slot. $K = 4$ and $N = 8$.

where $\epsilon = \epsilon_{\min} + (\epsilon_{\max} - \epsilon_{\min})e^{-C/\epsilon_{\text{decay}}}$, $\epsilon_{\min}$, $\epsilon_{\max}$, and ϵ_{decay} are constants, and C is the training iteration counter. The ϵ-greedy policy is adopted to avoid overfitting during the training of DNN modules. To determine the action $\boldsymbol{a}_n(t) = \arg\max_{\boldsymbol{a}_n \in \mathcal{A}_n} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}(t), \boldsymbol{a}_n)$ in the current time slot t, user n feeds the state $\boldsymbol{s}(t)$ into the DNN module, and determines the pilot sequence selection in the current time slot t based on the output of the DNN module. The proposed online algorithm is shown in Algorithm 2.

IV. Performance Evaluation

In this section, we present simulation results for the proposed scheme and compare them with the optimal scheme, the ACK-based scheme [4], and the random selection scheme. We use PyTorch [15] for simulation. The optimal scheme is the centralized solution of problem (5), where the pilot sequence selection is determined in a centralized manner, such that there is no collision for pilot sequence selection. In the random selection scheme, each user randomly selects one pilot

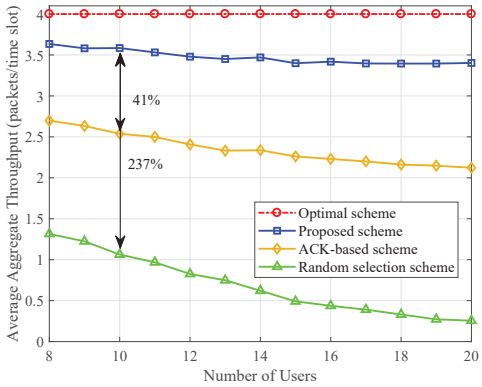

Fig. 5. Average aggregate throughput versus the number of users. $K = 4$.

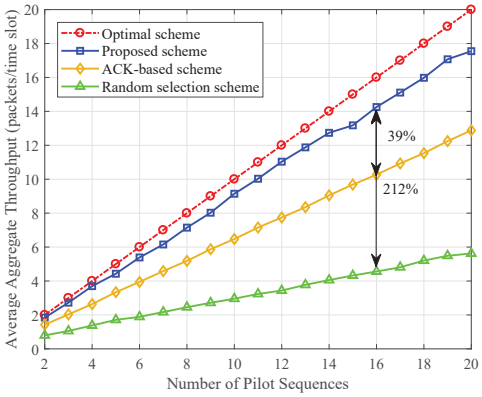

Fig. 6. Average aggregate throughput versus the number of pilot sequences.

sequence. In simulations, the radius of the circular cell is 800 m. We set the maximum transmit power $P^{\max}$ to 23 dBm, P_0 for open loop power control is -90 dBm [1]. The path loss exponent β is set to 4. For the LSTM layer, we set H to 64. The *size* (i.e., the maximum number of system transition tuples that can be stored) of the long-term replay at the base station is set to 4000, while each user maintains a short-term replay of size 20. For the ϵ-greedy policy, we set $\epsilon_{\min}$ to 0.05, $\epsilon_{\max}$ to 0.95, and ϵ_{decay} to 200. For the SGD algorithm, we set α to 0.01. For deep Q-learning, we set γ to 0.995 and M to 5. In each time slot, the base station trains the DNN modules for 20 iterations.

Fig. 4 shows the evolution of the average aggregate throughput across the time slot. The proposed scheme achieves 90% of the optimum after 80 time slots. A gap exists between the aggregate throughput of the proposed algorithm and the optimal scheme due to the errors in the Q-value approximation. The proposed scheme achieves an aggregate throughput which is 35% and 165% higher than that of the ACK-based scheme and the random selection scheme, respectively.

Fig. 5 shows the impact of the number of users on the average aggregate throughput when the number of pilot sequences K is equal to 4. The performance of the proposed scheme is evaluated after training for 150 time slots. When the number of users is equal to 10, the proposed scheme can achieve an aggregate throughput which is 41% and 237% higher than the ACK-based and random selection schemes, respectively.

Fig. 6 illustrates the average aggregate throughput versus the number of pilot sequences K, where we set the number of users to $2K$. The performance of the proposed scheme is evaluated after training for 150 time slots. While the average aggregate throughput of the other schemes increase with K, the proposed scheme can achieve an average aggregate throughput which is 39% and 212% higher than that of the ACK-based scheme and the random selection scheme, respectively, when the number of pilot sequences is equal to 16.

V. Conclusion

In this paper, we proposed a distributed pilot sequence selection algorithm for improving the aggregate throughput in GF-NOMA systems. We first formulated a pilot sequence selection problem for throughput optimization. We then designed a DRL-based distributed pilot sequence selection algorithm, which enables the users to gain experience from their action history and improve their pilot sequence selection. Simulation results showed that the proposed scheme can achieve a significant higher aggregate throughput than two previously reported GF-NOMA schemes. In our future work, we will consider throughput optimization for GF-NOMA systems where the users are equipped with multiple antennas.

References

[1] 3GPP TR 38.812 V16.0.0, "Technical specification group radio access network; Study on non-orthogonal multiple access (NOMA) for NR (Release 16)," Dec. 2018.

[2] V. W. S. Wong, R. Schober, D. W. K. Ng, and L. Wang, *Key Technologies for 5G Wireless Systems*. Cambridge University Press, 2017.

[3] L. Dai, B. Wang, Z. Ding, Z. Wang, S. Chen, and L. Hanzo, "A survey of non-orthogonal multiple access for 5G," *IEEE Commun. Surveys & Tuts.*, vol. 20, no. 3, pp. 2294–2323, Third Quarter 2018.

[4] S. Han, X. Tai, W. Meng, and C. Li, "A resource scheduling scheme based on feed-back for SCMA grant-free uplink transmission," in *Proc. IEEE Int'l Conf. on Commun. (ICC)*, Paris, France, May 2017.

[5] J. Shen, W. Chen, F. Wei, and Y. Wu, "ACK feedback based UE-to-CTU mapping rule for SCMA uplink grant-free transmission," in *Proc. Int'l Conf. Wireless Commun. Signal Process.*, Nanjing, China, Oct. 2017.

[6] J. Sun, W. Wu, and X. Wu, "A contention transmission unit allocation scheme for uplink grant-free SCMA systems," in *Proc. IEEE Int'l Conf. on Commun. (ICC)*, Kansas City, MO, May 2018.

[7] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[8] S. Kapturowski, G. Ostrovski, W. Dabney, J. Quan, and R. Munos, "Recurrent experience replay in distributed reinforcement learning," in *Proc. Int'l Conf. Learn. Representations*, New Orleans, LA, May 2019.

[9] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, pp. 257–265, Jun. 2018.

[10] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, Jan. 2019.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.

[12] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, 2nd ed., G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Springer, 2012, pp. 421–436.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[14] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. on Artificial Intelligence*, Phoenix, AZ, Feb. 2016.

[15] A. Paszke *et al.*, "PyTorch," 2018. [Online]. Available: https://pytorch.org/