Throughput Optimization for Grant-Free Multiple Access with Multiagent Deep Reinforcement Learning

Rui Huang, *Student Member, IEEE*, Vincent W.S. Wong, *Fellow, IEEE*, and Robert Schober, *Fellow, IEEE*

***Abstract*—Grant-free multiple access (GFMA) is a promising paradigm to efficiently support uplink access of Internet of Things (IoT) devices. In this paper, we propose a deep reinforcement learning (DRL)-based pilot sequence selection scheme for GFMA systems to mitigate potential pilot sequence collisions. We formulate a pilot sequence selection problem for aggregate throughput maximization in GFMA systems with specific throughput constraints as a Markov decision process (MDP). By exploiting multiagent DRL, we train deep neural networks (DNNs) to learn near-optimal pilot sequence selection policies from the transition history of the underlying MDP without requiring information exchange between the users. While the training process takes advantage of global information, we leverage the technique of factorization to ensure that the policies learned by the DNNs can be executed in a distributed manner. Simulation results show that the proposed scheme can achieve an average aggregate throughput that is within 85% of the optimum, and is 31%, 128%, and 162% higher than that of acknowledgement-based GFMA, dynamic access class barring, and random selection GFMA, respectively. Our results also demonstrate the capability of the proposed scheme to support IoT devices with specific throughput requirements.**

***Index Terms*—Grant-free multiple access (GFMA), deep reinforcement learning (DRL), medium access control (MAC) protocols, Internet of Things (IoT).**

I. Introduction

Internet of Things (IoT) is a promising paradigm that supports various types of applications, including smart home, smart city, intelligent transportation systems, and eHealth [2]. Hence, IoT has been recognized as an important enabler of Industry 4.0 [3]. While IoT benefits from the data provided by a large number of devices [4], this also introduces challenges to wireless communication systems. With limited resources, wireless communication systems are required to support a tremendous number of IoT devices. It is estimated that by 2023, there will be 14.7 billion machine-type connections globally [5]. Besides, different IoT devices may have different data rate or throughput requirements depending on their applications and services [2]. Hence, wireless communication systems have to enable efficient IoT data transmission such that a large number of devices with different throughput requirements can be supported.

Grant-free multiple access (GFMA) can tackle these emerging challenges in wireless systems by enhancing spectrum efficiency and reducing access delay [6], [7]. In GFMA, an IoT device selects a pilot sequence from a pre-allocated resource pool, and transmits its data to the base station without sending an access request to the base station *a priori*. The base station sends an acknowledgement (ACK) to the device upon successful decoding. The timing sequence diagram for GFMA is illustrated in Fig. 1. Compared with the four-step grant-based random access in Long Term Evolution (LTE), grant-free multiple access has a two-step access procedure. Hence, GFMA incurs a lower signaling overhead and reduces the access delay of IoT devices during the random access procedure. Moreover, when combined with non-orthogonal multiple access (NOMA), multiple IoT devices can transmit their packets simultaneously to the base station sharing the same physical resource block (PRB) in GFMA systems [8].

To fully exploit the benefits of GFMA, two challenges have to be overcome. First, due to the lack of centralized scheduling, packet collisions occur when multiple IoT devices select the same pilot sequence, which can lead to decoding failure and throughput degradation. Therefore, each device should choose a specific pilot sequence that distinguishes its signal from the signals of other devices to ensure successful channel estimation and decoding at the receiver [8]. Second, heterogeneous IoT devices cannot coordinate their transmissions or exchange information with each other. Each device selects a pilot sequence independently without knowing the selection decisions of the other devices. Moreover, due to the lack of knowledge of the throughput requirements of the other devices, an IoT device may greedily occupy too many network resources such that the throughput requirements of the other devices cannot be satisfied.

Various schemes have been proposed to resolve collisions in the pilot sequence selection in GFMA systems [9]–[11]. The authors in [9] propose an ACK-based scheduling scheme, where a device that experienced a packet collision selects a new pilot sequence from the remaining pilot sequences, which have not yet been selected by other devices, and retransmits

Manuscript received on Feb. 20, 2020; revised on Jul. 21, 2020; accepted on Sep. 5, 2020. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). This paper has been published in part in the *Proceedings of the IEEE Global Communications Conference (Globecom)*, Waikoloa, HI, Dec. 2019 [1]. The editor coordinating the review of this paper and approving it for publication was Chuan Huang. *(Corresponding author: Vincent W.S. Wong.)*

R. Huang and V. W.S. Wong are with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, V6T 1Z4, Canada. (e-mail: {ruihuang, vincentw}@ece.ubc.ca).

R. Schober is with the Institute for Digital Communications, Friedrich-Alexander University of Erlangen-Nuremberg, Erlangen 91058, Germany (e-mail: robert.schober@fau.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

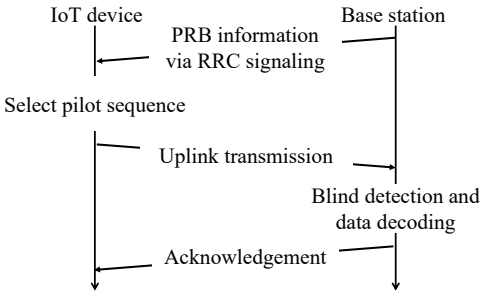


Fig. 1. The timing sequence diagram for GFMA. The base station configures the PRB via radio resource control (RRC) signaling [6]. Each IoT device then selects a pilot sequence and transmits its packet to the base station. After decoding, the base station informs the device about the decoding state by sending an ACK.

the packet. The authors in [10] propose that the base station reserves some of the pilot sequences for the retransmissions of devices that have suffered a packet collision. The base station then informs the devices about the reserved pilot sequences by broadcasting an ACK. For ACK-based solutions [9], [10], although collisions are resolved in the retransmission phase, collisions can still occur when a device transmits a packet for the first time as scheduling is performed only after a collision has occurred. The authors in [11] propose a pilot sequence allocation scheme, where the base station pre-assigns pilot sequences to those devices that have higher probabilities of transmitting packets in the next time slot. However, the allocation scheme in [11] requires centralized scheduling and accurate estimation of the transmit probability. In addition, the aforementioned schemes do not take into account any throughput requirements, and therefore may not be able to support different types of IoT applications.

Deep reinforcement learning (DRL) is a model-free learning technique based on deep neural networks (DNNs). DRL does not rely on a pre-established system model and is a powerful tool for solving optimization problems with large decision spaces [12], [13]. In contrast to multi-armed bandit problem based solutions [14]–[16], DRL-based solutions can be generalized and applied to problems with different system models and objectives. For the problem at hand, using DRL, a DNN can learn a distributed policy for pilot sequence selection by exploiting the pilot sequence selection history for training. Based on the pre-trained DNN, a device may avoid collisions with other devices without requiring information exchange between the devices.

By combining conventional multiagent reinforcement learning (MARL) with DNNs, multiagent DRL (MA-DRL) can efficiently handle nonstationary multiagent decision processes by jointly training multiple DNNs to jointly learn the policies [17]–[19]. Nevertheless, to the best of the authors' knowledge, the application of DRL for the design of GFMA protocols has not been considered yet. However, some insights can be obtained from the application of DRL for distributed spectrum access in cognitive radio [20]–[22]. The authors in [20] used DRL to design a distributed multi-channel access scheme to reduce the collision probability and maximize the channel utilization. DRL was employed to study cooperative and non-cooperative channel access of multiple users in [21]. The authors in [21] showed that, by properly designing a cooperative reward function, DRL-based channel access schemes may yield a better performance in terms of proportional fairness compared with using individual reward functions. The authors in [22] proposed a DRL-based channel access scheme for heterogeneous wireless networks. The results in [22] showed that, by using an MA-DRL framework along with a cooperative reward function, the maximum aggregate throughput and proportional fairness can be achieved. While the aggregate throughput maximization and proportional fairness, which are two special cases of α-fairness, have been investigated in [21], [22], the DRL frameworks proposed in the aforementioned studies cannot be applied to wireless systems where the users may have different throughput requirements. The latter case can be regarded as a generalized case of proportional fairness, where the users are prioritized based on their specific throughput requirements. Compared with α-fairness, in this case, it is more difficult for the DNNs to learn the pilot sequence selection policies with respect to the specific throughput requirements of the devices.

To address the aforementioned issues, in this paper, we propose an MA-DRL based distributed pilot sequence selection scheme for aggregate throughput maximization in GFMA systems, where we take into account different user throughput requirements. Each IoT device knows neither the pilot sequence selection decisions nor the throughput requirements of the other devices. Due to the lack of global information in each device, designing a distributed scheme that fosters collaboration between IoT devices such that the throughput requirements of different devices can be satisfied is challenging. In this paper, we exploit the potential of recurrent neural networks (RNNs) to handle the incomplete information of the underlying decision process and investigate their capability to learn the pilot sequence selection policies such that the aggregate throughput is maximized while the different throughput requirements are satisfied.

To be able to better tackle the nonstationarity and lack of coordination between devices, we further propose an MA-DRL training framework, in which the DNNs of all users are jointly trained to learn the pilot sequence selection policies with the help of global information at the base station. Using the technique of *factorization* [18], [23], the policies learned during joint training can be executed in a distributed manner. Our contributions are as follows:

- We formulate the pilot sequence selection problem for throughput optimization in GFMA systems with average throughput constraints for the IoT devices. We apply stochastic network optimization [24] and propose an algorithm to obtain the optimal solution. While relying on centralized scheduling, the optimal solution serves as a benchmark when evaluating the performance of the proposed DRL-based scheme.
- We model the pilot sequence selection process as a Markov decision process (MDP) and propose a pilot sequence selection scheme, where the DNNs are trained based on DRL to learn the pilot sequence selection policies from the transition history of the underlying

MDP for the IoT devices. We propose a deep recurrent Q-network (DRQN) to take advantage of the capability of RNNs to efficiently learn the temporal correlations of system transitions in time series learning problems.
- Using the factorization technique, we propose a *centralized training distributed execution* framework. In the proposed training framework, the DRQNs of all IoT devices are jointly trained by leveraging global information at the base station, whereas the policies learned during the centralized training phase can be executed in a distributed online manner.
- We conduct simulations to evaluate the performance of the proposed scheme. Our results show that DNNs are capable of learning near-optimal pilot sequence selection policies for IoT devices. For the considered system, the proposed scheme can achieve an aggregate throughput that is within 85% of the optimum. The aggregate throughput of the proposed scheme is 31%, 128%, and 162% higher than that of an ACK-based scheme [9], dynamic access class barring (ACB) [25], and a random selection scheme, respectively. The pilot sequence selection policies learned via DRL can also accommodate different throughput requirements of the IoT devices. Hence, the proposed scheme is capable of supporting different IoT applications in GFMA systems.

The remainder of this paper is organized as follows. The system model and problem formulation are introduced in Section II. The DRL framework for pilot sequence selection is presented in Section III. The DNN architectures and training algorithm are presented in Section IV. Simulation results are provided in Section V. Conclusions are drawn in Section VI.

II. System Model and Problem Formulation

We consider a GFMA system with one base station serving multiple users[1]. 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 $t \in \mathcal{T} = \{0, 1, 2, \ldots, T-1\}$. The base station proactively controls the network resource allocation for GFMA transmission in the considered system. In particular, in each time slot, the base station assigns one PRB, i.e., one time-frequency resource block, for GFMA transmission. We assume the base station assigns K pilot sequences to the PRB 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.

There are in total N users in the considered GFMA system. The set of users is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. All data packets are transmitted using GFMA. At the beginning of each time slot, the base station informs the users about the PRB and the K available pilot sequences via radio resource control (RRC) signaling. When a user decides to transmit, it selects one of the K available pilot sequences and performs uplink transmission. We assume that a user always has packets to send when it decides to perform uplink transmission. We define binary variable $g_{nk}(t) \in \{0, 1\}$, where $g_{nk}(t)$ is equal to 1 if user $n \in \mathcal{N}$ 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 $\boldsymbol{g}_n(t) \triangleq (g_{n1}(t), g_{n2}(t), \ldots, g_{nK}(t))$ as the pilot sequence selection vector of user n in time slot t. We define $n_k(t)$ as the number of users that select the k-th pilot sequence in time slot t. We have

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

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,

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

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

At the receiver side, the base station estimates the channels based on the received pilot sequences and then decodes the packets of the users. For the users who select a pilot sequence that is not chosen by other users, we assume the base station can perform perfect channel estimation and apply multiuser detection to mitigate the interference between the users [8], [26] and decode their packets successfully. However, when multiple users select the same pilot sequence, the base station cannot estimate the channels of the users, and hence cannot decode their packets [27]. In other words, 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 assume

$$r_n(t) \triangleq \begin{cases} 1, & \text{if } n \in \mathcal{S}(t), \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The base station sends an ACK to the user if its packet has been successfully decoded.

We use the time average expectation of the number of packets that are successfully received by the base station as the performance metric to measure the average aggregate throughput of the considered GFMA system. This metric has also been adopted in [22]. Since there are K pilot sequences, the maximum aggregate throughput of the considered system is K packets per time slot, which is the achievable aggregate throughput of the considered system in the absence of pilot sequence selection collisions. We denote $\mu_n(t)$ as the average throughput of user n up to time slot t. We have

$$\mu_n(t) \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}[r_n(\tau)], \quad n \in \mathcal{N},\, t \in \mathcal{T} \setminus \{0\}, \tag{4}$$

and $\mu_n(0) \triangleq 0$, where $\mathbb{E}[\cdot]$ is the expectation with respect to the randomness of the pilot sequence selections of the other

[1]In the remainder of this paper, we use the terms *users* and *IoT devices* interchangeably.

users. Given (4), we have $\mu_n(t) \in [0,1]$ for $n \in \mathcal{N}$, $t \in \mathcal{T}$. Moreover, we assume that each user knows its own average throughput but not that of the other users. This has two main reasons. First, we consider the average throughput of a particular user to be private information. Hence, from the perspective of the mobile network operator, the base station should not broadcast the average throughput of the users in order to avoid the potential leakage of private information. Second, broadcasting the average throughput of the users incurs an additional signaling overhead, which scales with the number of users K.

We denote user n's average required throughput by μ_n^{req}. Hence, the constraint with respect to the average throughput requirement of user n is given by

$$\limsup_{T\to\infty} \mu_n(T) = \limsup_{T\to\infty} \frac{1}{T}\sum_{\tau=0}^{T-1} \mathbb{E}[r_n(\tau)] \geq \mu_n^{\text{req}}, \quad n \in \mathcal{N}. \tag{5}$$

We assume that the average required throughputs of the IoT applications can be categorized into different quality of service (QoS) levels, and each of the QoS levels is pre-assigned with a QoS flow identifier (QFI), which is a positive integer [28, Section 5.7]. When sending a packet to the base station, a user indicates its QFI in the packet header, and the base station is informed about the average required throughput of the user by checking the packet header [28, Section 5.8].

Our objective is to maximize the average aggregate throughput in GFMA systems subject to user-specific throughput constraints. This leads to the following optimization problem

$$\begin{aligned} \underset{\boldsymbol{g}_n(t),\, n\in\mathcal{N},\, t\in\mathcal{T}}{\text{maximize}} \quad & \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{n\in\mathcal{N}} \mathbb{E}[r_n(t)] \\ \text{subject to} \quad & \sum_{k\in\mathcal{K}} g_{nk}(t) \leq 1,\, n \in \mathcal{N},\, t \in \mathcal{T}, \\ & \limsup_{T\to\infty} \frac{1}{T}\sum_{\tau=0}^{T-1} \mathbb{E}[r_n(\tau)] \geq \mu_n^{\text{req}},\, n \in \mathcal{N}. \end{aligned} \tag{6}$$

Problem (6) is a combinatorial optimization problem that can be solved using stochastic network optimization with virtual queues [24]. The details of this approach can be found in the Appendix. However, the optimal solution of problem (6) requires global information and centralized computation. Besides, centralized scheduling is necessary for implementing the optimal solution of problem (6) in GFMA systems. Hence, in this paper, we propose a practical DRL-based online pilot sequence selection scheme to obtain a suboptimal solution of problem (6) in a distributed manner.

III. MA-DRL Framework for GFMA Systems

With the recent advances in machine learning and artificial intelligent, DRL has emerged as a powerful tool for developing solutions for combinatorial optimization problems. In this section, we model the pilot sequence selection problem as an MDP and propose an MA-DRL based framework to solve problem (6) efficiently.

A. Modeling with MDP

As the pilot sequence selection of the N users in the considered GFMA system is essentially a multiagent decision making process, we model it as an MDP, which can be defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. The details are as follows:

1) State $\mathcal{S}$: We define variable $d_k(t) \in \{-1, 0, 1\}$, $k \in \mathcal{K}$, as the indicator for the decoding state of the k-th pilot sequence at the beginning of the current time slot $t \in \mathcal{T}$. $d_k(t)$ depends on the pilot sequence selection of the users in the previous time slot $t-1$. Specifically, $d_k(t)$ is equal to 1 if the signal using the k-th pilot sequence was decoded successfully in time slot $t-1$. We set $d_k(t)$ to -1 if the signal using the k-th pilot sequence was not decoded successfully in time slot $t-1$. $d_k(t)$ is equal to 0 if no signal using the k-th pilot was transmitted in time slot $t-1$. In time slot t, the *global* state $\boldsymbol{s}(t)$ consists of the decoding states of all pilot sequences and the average throughput of all users up to time slot t. We have

$$\boldsymbol{s}(t) \triangleq (d_1(t), \ldots, d_K(t), \mu_1(t), \ldots, \mu_N(t)), \quad t \in \mathcal{T}, \tag{7}$$

where $\mu_n(t)$ is given in (4). After decoding the packets of all users, the base station knows the decoding states of all pilot sequences $d_1(t), \ldots, d_K(t)$, and it can track the average throughput of all users $\mu_1(t), \ldots, \mu_N(t)$. Therefore, global state $\boldsymbol{s}(t)$ is available at the base station in each time slot. We define $\mathcal{S} \subseteq \{-1, 0, 1\}^K \times [0,1]^N$ to be the set of all the possible global states.

We assume the base station includes the information on $d_1(t), \ldots, d_K(t)$ in the RRC signaling and broadcasts this information to the users at the beginning of time slot t. Each $d_k(t)$, $k \in \mathcal{K}$ can be encoded using 2 bits. Hence, for a GFMA system with K pilot sequences, this incurs an overhead of $2K$ bits for downlink signaling. However, apart from knowing the decoding states of all K available pilot sequences, user n only knows its average throughput $\mu_n(t)$. Therefore, we define the *local* state of user n in time slot t as

$$\boldsymbol{s}_n(t) \triangleq (d_1(t), \ldots, d_K(t), \mu_n(t)), \quad t \in \mathcal{T}. \tag{8}$$

We define $\widehat{\mathcal{S}} \subseteq \{-1, 0, 1\}^K \times [0,1]$ as the set of all possible local states of user $n \in \mathcal{N}$. We note that as the state includes the average throughput of the users, the state in one particular time slot depends on the state of the previous time slot and the pilot sequence selections of the users.

2) Joint Action $\mathcal{A}$: In time slot t, the action profile of user n is its own pilot sequence selection $\boldsymbol{g}_n(t)$. To reduce the dimensionality of the action space, we use the index of the pilot sequence selected by user n in time slot t to indicate its action. That is

$$a_n(t) \in \widehat{\mathcal{A}} = \{0, 1, \ldots, K\}, \quad n \in \mathcal{N},\, t \in \mathcal{T}, \tag{9}$$

where $a_n(t) = 0$ means user n does not transmit in time slot t. The joint action of all users in time slot t is given by

$$\boldsymbol{a}(t) \triangleq (a_1(t), \ldots, a_N(t)),\, t \in \mathcal{T}. \tag{10}$$

The joint action space $\mathcal{A}$ is equal to $\widehat{\mathcal{A}}^N$.

3) State Transition Probability $\mathcal{P}$: The state transition probability $\mathbb{P}(\boldsymbol{s}(t+1)\,|\,\boldsymbol{s}(t),\boldsymbol{a}(t)),\ t\in\mathcal{T}$ is the probability that given current state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$, the next state is $\boldsymbol{s}(t+1)$, where $\mathbb{P}(\cdot)$ denotes the probability of event $(\cdot)$. In the considered GFMA system, the next state $\boldsymbol{s}(t+1)$ can be determined with probability (w.p.) one if the current state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$ are given. We denote the state transition probabilities as $\mathcal{P}:\mathcal{S}\times\mathcal{S}\times\mathcal{A}\to[0,1]$.

4) Reward $\mathcal{R}$: The reward is a real number that is determined based on the state and joint action of all users. In the considered MDP, given the global state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$, all users receive an aggregate reward, which is defined as follows

$$R(\boldsymbol{s}(t),\boldsymbol{a}(t))\triangleq\sum_{n\in\mathcal{N}}\Big(\hat{r}_n(t)-\lambda_n(t)(\mu_n^{\text{req}}-\mu_n(t))\Big),\ t\in\mathcal{T}, \tag{11}$$

where the term $\hat{r}_n(t)$ is given by

$$r_n(t)=\begin{cases}1, & \text{the packet of user } n \text{ has been successfully received,}\\ -1, & \text{the packet of user } n \text{ collided with those of other users,}\\ 0, & \text{otherwise.}\end{cases} \tag{12}$$

The term $\lambda_n(t)(\mu_n^{\text{req}}-\mu_n(t))$ is the penalty resulting from violating the throughput constraint of user $n\in\mathcal{N}$. We set $\lambda_n(t)$ to a positive constant C when $\mu_n(t)<\mu_n^{\text{req}}$, and equal to zero when the throughput constraint is satisfied. That is,

$$\lambda_n(t)=\begin{cases}C, & \text{if } \mu_n(t)<\mu_n^{\text{req}},\\ 0, & \text{otherwise.}\end{cases} \tag{13}$$

We define $\mathcal{R}\subseteq\mathbb{R}$ to be the set of rewards.

B. Global Q-Value Approximation with MA-DRL

To maximize the aggregate reward in the considered MDP, the optimal joint action *may* be determined with conventional Q-learning [29]. In Q-learning, the expected cumulative discounted reward of taking the joint action $\boldsymbol{a}$ under state $\boldsymbol{s}$ is given by the *global Q-value* $Q^{\text{G}}(\boldsymbol{s},\boldsymbol{a})$, which is given by [29, Section 3.5]

$$Q^{\text{G}}(\boldsymbol{s},\boldsymbol{a})=\mathbb{E}\Bigg[\sum_{k=0}^{\infty}\gamma^k\ R(\boldsymbol{s}(t+k),\boldsymbol{a}(t+k))\ \Bigg|\ \boldsymbol{s}(t)=\boldsymbol{s},\ \boldsymbol{a}(t)=\boldsymbol{a}\Bigg], \tag{14}$$

where $\gamma\in[0,1]$ is the discount factor. The global Q-value is updated during the decision process. Given state $\boldsymbol{s}(t)$ in time slot t, the users choose joint action $\boldsymbol{a}(t)$, receive an aggregate reward $R(\boldsymbol{s}(t),\boldsymbol{a}(t))$, and the next state is $\boldsymbol{s}(t+1)$. Then, the global Q-value is updated as follows

$$Q^{\text{G}}(\boldsymbol{s}(t),\boldsymbol{a}(t))\leftarrow\mathbb{E}\Bigg[R(\boldsymbol{s}(t),\boldsymbol{a}(t))+\gamma\max_{\boldsymbol{a}'\in\mathcal{A}}Q^{\text{G}}(\boldsymbol{s}(t+1),\boldsymbol{a}')\ \Bigg|\ \boldsymbol{s}(t)=\boldsymbol{s},\ \boldsymbol{a}(t)=\boldsymbol{a}\Bigg]. \tag{15}$$

Estimating the global Q-value is beneficial in a multiagent decision process as the global state $\boldsymbol{s}(t)$ and joint action $\boldsymbol{a}(t)$ contain the information about all the users. Hence, the global Q-value can capture the impact of the action of a user on other users, and therefore can handle the nonstationary case of a multiagent decision process [30], [31]. To determine the action $\boldsymbol{a}(t)$ that maximizes the expected cumulative discounted reward, in conventional Q-learning, a Q-table is required to store the global Q-values of all possible actions for a given state $\boldsymbol{s}(t)$. However, the size of the Q-table increases with the cardinalities of the action and state spaces. This makes Q-learning costly in terms of computation and memory, especially for IoT devices.

Deep Q-learning [12] 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 a given state and the corresponding Q-values of all possible actions. A DNN module with learnable parameters $\boldsymbol{\Phi}$ can be employed to approximate the global Q-values. In fact, $\boldsymbol{\Phi}$ is a vector that collects the weights and biases of the neurons within the DNN module.

The learnable parameters $\boldsymbol{\Phi}$ are updated based on the system transition history to improve the accuracy of the Q-value approximation. This is referred to as the *training phase* of the DNN. In current time slot t, the system transition history consists of the *system transition tuples* $(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau),R(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau)),\boldsymbol{s}(\tau+1))$ 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 global Q-value for $\boldsymbol{s}(\tau)$ and $\boldsymbol{a}(\tau)$, which is approximated by a DNN with parameters $\boldsymbol{\Phi}$, as $Q^{\text{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))$. In each training iteration, we store the parameters of the DNN resulting from the previous training iteration, which we denote as $\widehat{\boldsymbol{\Phi}}$. Then, the target global Q-value is given by $R(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))+\gamma\max_{\boldsymbol{a}\in\mathcal{A}}Q^{\text{G}}_{\widehat{\boldsymbol{\Phi}}}(\boldsymbol{s}(\tau+1),\boldsymbol{a})$ [12]. The update of $\boldsymbol{\Phi}$ is determined by minimizing the temporal difference (TD) error between the target and the approximated global Q-value [12]. That is,

$$\arg\min_{\boldsymbol{\Phi}}\ \ \frac{1}{2}\big(R(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))+\gamma\max_{\boldsymbol{a}\in\mathcal{A}}Q^{\text{G}}_{\widehat{\boldsymbol{\Phi}}}(\boldsymbol{s}(\tau+1),\boldsymbol{a})-Q^{\text{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))\big)^2. \tag{16}$$

To solve problem (16), we apply a stochastic gradient descent (SGD) algorithm to update parameters $\boldsymbol{\Phi}$. Specifically, parameters $\boldsymbol{\Phi}$ are updated as follows [32]

$$\boldsymbol{\Phi}\leftarrow\boldsymbol{\Phi}-\alpha\,\nabla Q^{\text{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))\,\big(R(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))+\gamma\max_{\boldsymbol{a}\in\mathcal{A}}Q^{\text{G}}_{\widehat{\boldsymbol{\Phi}}}(\boldsymbol{s}(\tau+1),\boldsymbol{a})-Q^{\text{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))\big), \tag{17}$$

where α is the learning rate. The gradient $\nabla Q^{\text{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))$ in (17) is determined using the backpropagation algorithm [33, Ch. 6]. After updating parameters $\boldsymbol{\Phi}$ with a sufficient number of system transition tuples, the optimal joint action can be determined based on the pre-trained DNN as follows

$$\boldsymbol{a}(t)=\arg\max_{\boldsymbol{a}\in\mathcal{A}}Q^{\text{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(t),\boldsymbol{a}). \tag{18}$$

To obtain the optimal joint action based on (18), we feed $\boldsymbol{s}(t)$ into the DNN and determine $\boldsymbol{a}(t)$ based on the output of the DNN.

C. Local Q-Value based on Factorization

To estimate the global Q-value, it is necessary for the users to know the joint action $\boldsymbol{a}(t)$ and the global state $\boldsymbol{s}(t)$ [30], [31]. However, as communication (or coordination) between the users in the considered GFMA system is not possible, a user cannot obtain knowledge about the actions of the other users or their local states to estimate the global Q-value directly. This means that an MARL algorithm that relies solely on the global Q-value cannot be implemented in the considered GFMA system in a distributed manner.

To tackle the aforementioned difficulties, rather than directly estimating the global Q-value, we propose that each user maintains a *local DNN module* to estimate the impact of its own action on the global Q-value. With the local DNN module, which is characterized by parameters $\boldsymbol{\Phi}_n$, each user $n \in \mathcal{N}$ can obtain a *local Q-value* denoted by $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))$ based on its local information, i.e., its action $a_n(\tau)$ and its local state $\boldsymbol{s}_n(\tau)$.

To ensure that the local Q-value estimated by the local DNN module of user n can capture the impact of user n's action on the global Q-value, a mapping between the global Q-value and the local Q-value should be learned. To obtain such a mapping, we adopt the idea of *factorization* in MARL and multiagent systems [23], [34], [35], according to which, for a given state, an action of a user that leads to a larger global Q-value should also result in a larger local Q-value. In particular, let $\boldsymbol{a}_{-n}(\tau)$ denote the actions of all users in set $\mathcal{N}$ except for user n in time slot τ. Given the global state $\boldsymbol{s}(\tau)$ and the local state $\boldsymbol{s}_n(\tau)$, the local Q-value approximated by the local DNN module of user n is a factorization of the global Q-value if [34, Ch. 3]

$$\begin{aligned} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau)) > \; & Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a'_n(\tau)) \\ \Leftrightarrow Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), (a_n(\tau), \boldsymbol{a}_{-n}(\tau))) \\ > \; & Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), (a'_n(\tau), \boldsymbol{a}_{-n}(\tau))), \end{aligned} \tag{19}$$

where $a_n(\tau), a'_n(\tau) \in \widehat{\mathcal{A}}$ and $\boldsymbol{a}_{-n}(\tau) \in \widehat{\mathcal{A}}^{N-1}$. To obtain such a factorization, we use a monotonic function $F(\cdot)$ such that

$$\begin{aligned} Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = F\big(&Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1(\tau)), \\ &\cdots, Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N(\tau))\big). \end{aligned} \tag{20}$$

The monotonic function in (20) should satisfy the following condition

$$\frac{\partial F\big(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1(\tau)), \cdots, Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N(\tau))\big)}{\partial Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))} > 0. \tag{21}$$

Any monotonic function $F(\cdot)$ that satisfies constraint (21) guarantees that the resulting local Q-value is a factorization of the global Q-value as specified in (19).

We emphasize that the factorization of the global Q-value is the key to the design of a distributed pilot sequence selection scheme based on MA-DRL. The factorization of the global Q-value in (20) leads to the following property

$$\begin{aligned} \arg\max_{\boldsymbol{a}\in\mathcal{A}} Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = \Big(&\arg\max_{a_1\in\widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1), \\ &\ldots, \arg\max_{a_N\in\widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N)\Big). \end{aligned} \tag{22}$$

Equation (22) shows that the joint action that maximizes the global Q-value corresponds to the individual actions that maximize the local Q-value of each user $n \in \mathcal{N}$. In other words, each user can determine its own action that maximizes the global Q-value by greedily selecting the action that maximizes its local Q-value, which is

$$a_n(t) = \arg\max_{a_n\in\widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(t), a_n). \tag{23}$$

To obtain $a_n(t)$ in (23), user n feeds $\boldsymbol{s}_n(t)$ into its DNN and determines $a_n(t)$ based on the output of the DNN. This does not require knowledge of the actions of the other users or the global state, and therefore can be implemented in a distributed manner.

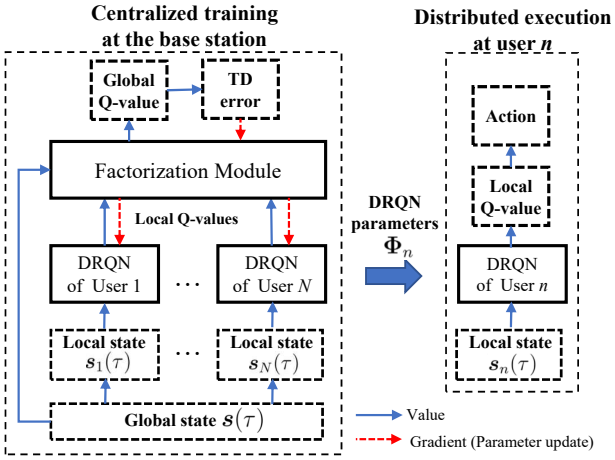


Fig. 2. The overall network architecture. The DRQNs of all users are jointly trained at the base station. The base station sends the updated parameters of the pre-trained DRQNs to the users. The users can then select pilot sequences in a distributed manner. The forward propagations for determining the Q-values and the TD-error are denoted by solid blue arrows, while the backpropagations for updating the learnable parameters are denoted by dashed red arrows.

IV. DNN Architecture and Proposed Scheme

In this section, we propose an architecture for the DNN modules to approximate the global and local Q-values. We also present an online pilot sequence selection scheme, in which the pilot sequence selections are determined based on the outputs of pre-trained DNN modules.

A. Overall Network Architecture

The overall network architecture of the proposed DNN module for approximating the global Q-value is illustrated in Fig. 2. The DNN module consists of two parts:

- Deep Recurrent Q-Network (DRQN): DRQN is a recurrent neural network (RNN)-based DNN module that can aggregate experience from the system transition history. We use DRQNs to generate the Q-values based on the states of the users.
- Factorization Module: The factorization module is a multi-layer perceptron (MLP) module which guarantees that the local Q-values of the users are the factorization of the approximated global Q-value. To this end, the factorization module is trained to approximate the monotonic function $F(\cdot)$ in (20) by taking advantage of the global information, i.e., the global state $\boldsymbol{s}(t)$.

In particular, each user maintains one local DRQN to determine the approximated local Q-value (right-hand side of Fig. 2), based on which the user can select its pilot sequence. Meanwhile, the training of the DRQNs of all users is performed by the base station. During the centralized training phase at the base station (left-hand side of Fig. 2), the DRQNs of all users are jointly trained based on global information. The advantages of using centralized training in wireless systems are two-fold. First, the base station, which is equipped with powerful hardware, can efficiently train the DNNs for the users. Second, the users (i.e., IoT devices) can prolong their battery lifetime by not being involved in the energy-consuming training phase. The factorization module is employed only during the training phase to ensure that the global Q-value can be properly factorized. After training, the base station sends the learnable parameters of the DRQNs to the users. The users select the pilot sequences based on the outputs of the DRQNs in a distributed manner without relying on global information. Note that the base station does not know which pilot sequence selections of the users lead to collisions. Moreover, the base station cannot differentiate between the users that remain silent and the users that suffer from collisions. This prevents the base station from knowing the joint action $\boldsymbol{a}(\tau)$. Hence, in practice, user n may include the history of its action $a_n(\tau)$ in its data packet, so that the base station can obtain the complete system transition history.

The proposed training framework falls into the category of *centralized training distributed execution* frameworks in distributed DRL and MA-DRL [13], [36]. We combine conventional MARL, in particular, the factorization technique, with DRL. The advantage of the proposed framework is that the distributed and user-specific pilot sequence selection policies can be learned during the centralized training process. Moreover, compared with conventional centralized MARL, the proposed learning framework is able to better handle the large joint action space with the help of DNNs. The details of the network architecture are presented in the following subsections.

B. DRQN Design

The DRQNs of all users have the same network architecture as illustrated in Fig. 3. For each user $n \in \mathcal{N}$, the DNN layers in the DRQN and their functionalities are as follows:

1) Input Layer: The input layer collects the local state and feeds it to the DNN. For the DRQN of user n, the input is the local state $\boldsymbol{s}_n(t)$, which is a vector of size $K+1$.

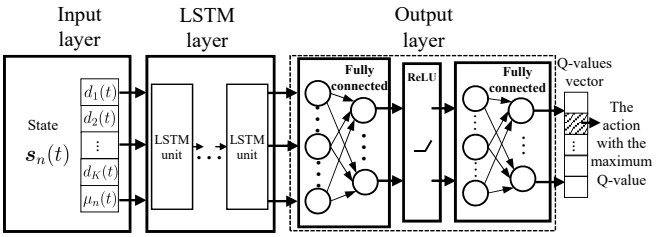


Fig. 3. The proposed network structure for DRQN, which consists of an input layer, an LSTM layer, and an output layer.

2) Long Short-Term Memory (LSTM) Layer: The local state $\boldsymbol{s}_n(t)$ collected by the input layer is fed into an LSTM layer. LSTM layer is a type of RNN. Using LSTM layer, the information about the system transition history can be learned and stored in its recurrent states. Therefore, LSTM layer can learn the temporal correlations of the system transitions more efficiently than feedforward neural networks. For the same reason, LSTM layer is capable of overcoming the lack of complete information in the underlying MDP [13]. In particular, we use an LSTM layer with D hidden units to aggregate the information from the system transition history and approximate the Q-values. The outputs of the LSTM layer are its hidden states.

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 generate the Q-values based on the hidden states of the LSTM layer. The output is a vector of size $K+1$. For $k \in \mathcal{K}$, the $(k+1)$-th entry of the output vector is the local Q-value for selecting the k-th pilot sequence in time slot t for local state $\boldsymbol{s}_n(t)$, while the first entry is the local Q-value for not transmitting in time slot t.

C. Factorization Module Design

The DNN architecture of the factorization module is shown in Fig. 4. To obtain the desired function $F(\cdot)$ for global Q-value factorization, we use an MLP network-based *universal approximator* (see the blue box in Fig. 4) to approximate function $F(\cdot)$, while the weights and biases of the MLP network are generated by four FC layers (see the green box in Fig. 4). As shown in [37], an MLP network with one hidden layer can serve as a universal approximator. In particular, the input layer of the MLP network has N neurons and each neuron takes respectively one of the local Q-values approximated by the DRQNs of the N users as input. The hidden layer of the MLP network has H neurons. There is one neuron in the output layer as the global Q-value is a scalar. We denote the weight between the j-th neuron of the hidden layer and the n-th neuron of the input layer and the bias of the j-th neuron of the hidden layer in time slot τ as $v_{jn}(\tau)$ and $b_j(\tau)$, respectively. We denote the weight between the j-th neuron of the hidden layer and the neuron of the output layer in time slot τ as $w_j(\tau)$. The bias of the neuron of the output layer in time slot τ is referred to as $b_0(\tau)$. Then, function $F(\cdot)$

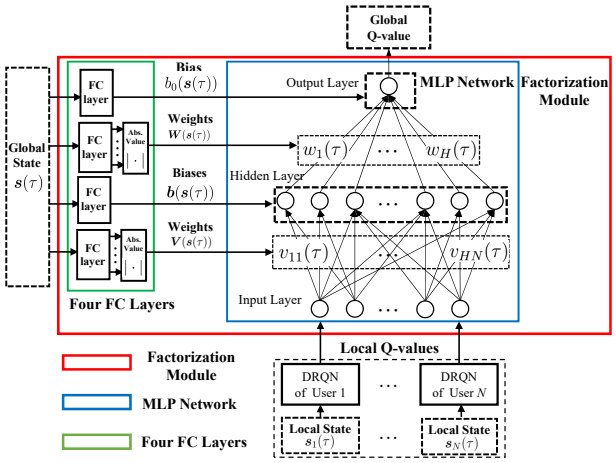


Fig. 4. DNN architecture for the factorization module located at the base station. The factorization module, which is indicated by the red box in the figure, consists of two parts: (i) a universal approximator for obtaining the monotonic function $F(\cdot)$, which is an MLP network with one hidden layer as indicated by the blue box in the figure, and (ii) four FC layers, i.e., the green box in the figure, for generating the weights and biases of the MLP network. Each FC layer in the factorization module takes the global state as input.

obtained with the MLP network can be expressed as follows [37]

$$F(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau),\boldsymbol{a}_1(\tau)),\ldots,Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau),\boldsymbol{a}_N(\tau))) = b_0(\tau)+\sum_{j=1}^{H} w_j(\tau)h\Big(b_j(\tau) + \sum_{n\in\mathcal{N}} v_{jn}(\tau)Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau),a_n(\tau))\Big), \quad (24)$$

where $h(\cdot)$ is the sigmoid function.

Recall that, in order to ensure that the local Q-value is a factorization of the global Q-value, function $F(\cdot)$ should satisfy condition (21). This means that function $F(\cdot)$ should be entry-wise monotonic increasing with respect to the inputs $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau),a_n(\tau))$, $n\in\mathcal{N}$. For the function approximated by the universal approximator (24), since the sigmoid function $h(\cdot)$ is monotonically increasing, condition (21) is satisfied if the weights, i.e., $w_j(\tau)$ and $v_{jn}(\tau)$, in time slot $\tau\in\mathcal{T}$ are all positive [37]. Hence, we emphasize that the key to obtaining a function $F(\cdot)$ which possesses the desired factorization property is to enforce the positivity of the weights of the MLP network during the training phase.

As the global Q-value $Q^{\mathrm{G}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))$ depends on the global state $\boldsymbol{s}(\tau)$, it is intuitive to let the approximated monotonic function $F(\cdot)$ depend on the global state as well. Therefore, we use DNNs to generate the weights and biases of the universal approximator in time slot τ based on the global state $\boldsymbol{s}(\tau)$. By doing this, we let the weights and biases of the universal approximator depend on the specific state in the time slot, so that the monotonic function $F(\cdot)$ can be adjusted with respect to different states, e.g., the different achieved throughputs of the users. This makes the universal approximator more flexible. To take the effect of state $\boldsymbol{s}(\tau)$ on the weights and biases into account, we replace τ in their arguments in (24) by $\boldsymbol{s}(\tau)$. The universal approximator (24) can then be written in the following matrix notation

$$F\big(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau),\boldsymbol{a}_1(\tau)),\ldots,Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau),\boldsymbol{a}_N(\tau))\big) = \boldsymbol{W}(\boldsymbol{s}(\tau))\, h\big(\boldsymbol{V}(\boldsymbol{s}(\tau))\,\boldsymbol{Q}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau)) + \boldsymbol{b}(\boldsymbol{s}(\tau))\big) + b_0(\boldsymbol{s}(\tau)), \quad (25)$$

where function $h(\cdot)$ in (25) is the element-wise sigmoid function and

$$\boldsymbol{Q}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau)) = \Big[Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau),\boldsymbol{a}_1(\tau)) \;\cdots\; Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau),\boldsymbol{a}_N(\tau))\Big]^{\top} \in \mathbb{R}^{N\times 1}, \quad (26)$$

$$\boldsymbol{V}(\boldsymbol{s}(\tau)) = \begin{bmatrix} v_{11}(\boldsymbol{s}(\tau)) & \cdots & v_{1N}(\boldsymbol{s}(\tau)) \\ \vdots & \ddots & \vdots \\ v_{H1}(\boldsymbol{s}(\tau)) & \cdots & v_{HN}(\boldsymbol{s}(\tau)) \end{bmatrix} \in \mathbb{R}^{H\times N}, \quad (27)$$

$$\boldsymbol{b}(\boldsymbol{s}(\tau)) = \big[b_1(\boldsymbol{s}(\tau))\; b_2(\boldsymbol{s}(\tau))\; \cdots\; b_H(\boldsymbol{s}(\tau))\big]^{\top} \in \mathbb{R}^{H\times 1}, \quad (28)$$

and

$$\boldsymbol{W}(\boldsymbol{s}(\tau)) = \big[w_1(\boldsymbol{s}(\tau))\; w_2(\boldsymbol{s}(\tau))\; \cdots\; w_H(\boldsymbol{s}(\tau))\big] \in \mathbb{R}^{1\times H}. \quad (29)$$

The weights $\boldsymbol{V}(\boldsymbol{s}(\tau))$, $\boldsymbol{W}(\boldsymbol{s}(\tau))$ and biases $\boldsymbol{b}(\boldsymbol{s}(\tau))$, $b_0(\boldsymbol{s}(\tau))$ in time slot τ are generated by multiple DNNs separately to improve the representational capacity of the universal approximator [19].

In particular, we use one FC layer to generate the weights $\boldsymbol{V}(\boldsymbol{s}(\tau))$. The input of this FC layer is the global state $\boldsymbol{s}(\tau)$, and the output is a vector of size HN. The output vector is then rearranged into an $H\times N$ matrix. The weights $\boldsymbol{V}(\boldsymbol{s}(\tau))$ are obtained by taking the absolute values of all elements of the output matrix. We use a similar method to obtain the weights $\boldsymbol{W}(\boldsymbol{s}(\tau))$ as well. Note that by taking the absolute values, we ensure that the weights in (24) are all positive, and therefore the desired factorization property is achieved. We employ one FC layer to obtain the biases $\boldsymbol{b}(\boldsymbol{s}(\tau))$. This FC layer takes $\boldsymbol{s}(\tau)$ as input, and the output is a vector $\boldsymbol{b}(\boldsymbol{s}(\tau))$ of size H. It is not necessary to make the biases positive as the signs of the biases will not affect the monotonicity. We use a similar method to obtain the bias $b_0(\boldsymbol{s}(\tau))$. The output dimension of the FC layer for $b_0(\boldsymbol{s}(\tau))$ is equal to 1.

D. Joint Training Algorithm

We use $\boldsymbol{\Phi}_0$ to denote the vector that contains the learnable parameters of the factorization module. Since the monotonic function $F(\cdot)$ in time slot τ depends on both the global state $\boldsymbol{s}(\tau)$ and the parameters $\boldsymbol{\Phi}_0$, we denote it as $F_{\boldsymbol{s}(\tau),\boldsymbol{\Phi}_0}(\cdot)$. Based on the proposed network architecture, the parameters $\boldsymbol{\Phi}$ used to approximate global Q-value $Q^{\mathrm{G}}_{\boldsymbol{\Phi}}(\boldsymbol{s}(\tau),\boldsymbol{a}(\tau))$ are now given by $\boldsymbol{\Phi}=(\boldsymbol{\Phi}_0,\boldsymbol{\Phi}_1,\ldots,\boldsymbol{\Phi}_N)$, which includes the parameters of the N DRQNs and the parameters of the factorization module. The training algorithm is summarized

Algorithm 1 Offline Training Algorithm for MA-DRL Framework in Each Training Iteration

1: Sample a minibatch of system transition tuples $(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1)), \tau \in \mathcal{T}_b'$.
2: **for** $n \in \mathcal{N}$ **do**
3: Determine $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))$ and $\max_{a_n \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n)$ with the DRQN of user n.
4: **end for**
5: Determine the approximated global Q-value $Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ based on (30).
6: Determine the target global Q-value $y_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)))$ based on (31).
7: Update the parameters $\boldsymbol{\Phi}$ using SGD (34).

in Algorithm 1. In Line 5, the approximated global Q-value for state $\boldsymbol{s}(\tau)$ and joint action $\boldsymbol{a}(\tau)$ is given by

$$Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) = F_{\boldsymbol{s}(\tau), \boldsymbol{\Phi}_0}\big(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau), a_1(\tau)), \ldots, Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau), a_N(\tau))\big). \quad (30)$$

To obtain $Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$, we first feed the local state $\boldsymbol{s}_n(\tau), n \in \mathcal{N}$, in time slot τ into the DRQN of user n. $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau))$ can be obtained by selecting the value of the entry of the output that corresponds to $a_n(\tau)$. Then, we feed the global state $\boldsymbol{s}(\tau)$ into the factorization module. With the outputs of the four FC layers, the weights $\boldsymbol{V}(\boldsymbol{s}(\tau)), \boldsymbol{W}(\boldsymbol{s}(\tau))$ and the biases $\boldsymbol{b}(\boldsymbol{s}(\tau)), b_0(\boldsymbol{s}(\tau))$ which yield the monotonic function $F_{\boldsymbol{s}(\tau), \boldsymbol{\Phi}_0}(\cdot)$. $Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))$ can now be obtained based on (25) with $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau), a_n(\tau)), n \in \mathcal{N}$, as inputs of the monotonic function.

The base station can now jointly train the factorization module and the DRQNs based on the TD-error of the global Q-value approximation. In Line 6, for the system transition tuple with time index τ, the target of the global Q-value approximation is

$$\begin{aligned} &y_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) \\ &= R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau+1), \boldsymbol{a}) \\ &= R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} F_{\boldsymbol{s}(\tau+1), \boldsymbol{\Phi}_0}\big(Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau+1), a_1), \\ &\qquad \ldots, Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau+1), a_N)\big) \\ &\overset{(a)}{=} R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)) + \gamma\, F_{\boldsymbol{s}(\tau+1), \boldsymbol{\Phi}_0}\big(\max_{a_1 \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau+1), a_1), \\ &\qquad \ldots, \max_{a_N \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau+1), a_N)\big). \end{aligned} \quad (31)$$

Equality (a) holds since given state $\boldsymbol{s}(\tau+1)$ and parameters $\boldsymbol{\Phi}_0$, the weights and biases in monotonic function $F_{\boldsymbol{s}(\tau+1), \boldsymbol{\Phi}_0}(\cdot)$ are constants. Given the local state $\boldsymbol{s}_n(\tau+1)$ and parameters $\boldsymbol{\Phi}_n$, the output of the DRQN of user n is a constant vector of size $K+1$, meaning that we only have $K+1$ options for the value of $Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n)$ for user n. As function $F_{\boldsymbol{s}(\tau+1), \boldsymbol{\Phi}_0}(\cdot)$ is entry-wise monotonically increasing, the maximum can be obtained by selecting the maximum element, $\max_{a_n \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n)$, in the output vector of user n's DRQN as the input of function $F_{\boldsymbol{s}(\tau+1), \boldsymbol{\Phi}_0}(\cdot)$.

To obtain the target global Q-value (31), we feed the local states of time slot $\tau+1$ into the DRQNs of the users accordingly and the value of $\max_{a_n \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n), n \in \mathcal{N}$, can be obtained by selecting the action that corresponds to the entry with the maximum value in the output. Then, we feed the global state $\boldsymbol{s}(\tau+1)$ into the factorization module. Based on the outputs of the four FC layers, we obtain the weights $\boldsymbol{V}(\boldsymbol{s}(\tau+1)), \boldsymbol{W}(\boldsymbol{s}(\tau+1))$ and biases $\boldsymbol{b}(\boldsymbol{s}(\tau+1)), b_0(\boldsymbol{s}(\tau+1))$, and determine the monotonic function $F_{\boldsymbol{s}(\tau+1), \boldsymbol{\Phi}_0}(\cdot)$. $F_{\boldsymbol{s}(\tau+1), \boldsymbol{\Phi}_0}(\max_{a_1 \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_1}(\boldsymbol{s}_1(\tau+1), a_1), \cdots, \max_{a_N \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_N}(\boldsymbol{s}_N(\tau+1), a_N))$ can now be obtained by using $\max_{a_n \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(\tau+1), a_n), n \in \mathcal{N}$, as the input of the monotonic function. Then, the TD-error for the global Q-value approximation with respect to the system transition tuple with time index τ is given by

$$\begin{aligned} &\mathcal{L}_G(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1)) \\ &= \frac{1}{2}\big(y_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) - Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))\big)^2. \end{aligned} \quad (32)$$

In a practical system, the base station maintains the system transition history, which is referred to as the *replay* in the DRL literature [12], [17]. In each training iteration, the base station samples a minibatch that contains multiple system transition tuples from the replay. The base station determines the TD-error in (32) for the tuples within the minibatch, and then updates the parameters $\boldsymbol{\Phi}$. To efficiently train the LSTM layer, instead of randomly sampling the system transition tuples from the long-term replay, we sample the system transition tuples of m consecutive time slots and feed the corresponding system transition history sequentially into the DNN module to update the parameters. By doing this, the hidden states of the LSTM layer can be carried forward throughout the entire training iteration to learn the temporal correlations from the system transition history. We note that training the LSTM layer with minibatches sampled from the replay is known to suffer from *initial recurrent state mismatch* [13], which may lead to inaccurate Q-value approximation. In particular, the minibatches sampled in two consecutive training iterations (i.e., the previous training iteration and the current training iteration) may represent the system transition history of two different time periods. This means the temporal correlations within the system transition history sampled in the previous training iteration may significantly differ from the ones sampled in the current training iteration. Therefore, the hidden states of the LSTM layer generated based on the system transition history sampled in the previous training iteration may not be able to accurately approximate the Q-values with respect to the system transition history sampled in the current training iteration.

To overcome the initial recurrent state mismatch, the hidden states of the LSTM layer should be initialized properly in each training iteration. We use the first l consecutive system transition tuples within the sampled minibatch to initialize the hidden states of the LSTM layer, so that the hidden states can be initialized based on the temporal correlations of the system

transition history sampled in the current training iteration (which is also referred to as the *burn-in method* [13]). This is accomplished by sequentially feeding the state history of the first l time slots into the DRQNs. Thus, the hidden states can be initialized and carried forward. Subsequently, we use the state history of the remaining $m-l$ consecutive time slots to update the learnable parameters. The TD-errors of global Q-value approximation are determined only for the states of the remaining $m-l$ time slots and then averaged. We use $\mathcal{T}_b'$ to denote the set of the time indices of the system transition tuples within the minibatch, excluding the first l transition tuples. For the sampled minibatch, the TD-error of the global Q-value approximation is

$$\mathcal{L}_G = \frac{1}{m-l} \sum_{\tau \in \mathcal{T}_b'} \mathcal{L}_G(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)), \boldsymbol{s}(\tau+1)). \tag{33}$$

Then, in Line 7, all learnable parameters $\boldsymbol{\Phi}$ are updated using SGD [18]:

$$\begin{aligned} \boldsymbol{\Phi} \leftarrow \ \boldsymbol{\Phi} - \frac{\alpha}{m-l} \sum_{\tau \in \mathcal{T}_b'} & (y_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau+1), R(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) \\ & - Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau))) \nabla Q_{\boldsymbol{\Phi}}^{\mathrm{G}}(\boldsymbol{s}(\tau), \boldsymbol{a}(\tau)). \end{aligned} \tag{34}$$

The training in (34) is an *end-to-end* learning process. That is, in each training iteration, the DRQNs of all users as well as the factorization module are jointly trained based on the TD-error of global Q-value approximation.

E. Proposed Online Scheme

As shown in (22) and (23), a user can determine its pilot sequence selection in a distributed manner by leveraging its local information and the pre-trained DRQN. In particular, the base station sends the parameters of user n's DRQN, i.e., $\boldsymbol{\Phi}_n$, back to user $n \in \mathcal{N}$. Each user maintains a local DRQN. Upon receiving the parameters $\boldsymbol{\Phi}_n$ from the base station, user n updates the parameters of its local DRQN to be the same as $\boldsymbol{\Phi}_n$. User n then applies an ϵ-greedy policy to select its pilot sequence. Specifically, the pilot sequence selection of user n in the current time slot t is determined by [12]

$$a_n(t) = \begin{cases} \arg\max\limits_{a_n \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(t), a_n), & \text{with probability } 1-\epsilon, \\ \text{random selection}, & \text{with probability } \epsilon, \end{cases} \tag{35}$$

where $\epsilon = \epsilon_{\min} + (\epsilon_{\max} - \epsilon_{\min}) e^{-G/\epsilon_{\text{decay}}}$, and $\epsilon_{\min}$, $\epsilon_{\max}$, and ϵ_{decay} are constants. G is the training iteration counter. The ϵ-greedy policy is adopted to avoid overfitting. To determine action $a_n(t) = \arg\max\limits_{a_n \in \widehat{\mathcal{A}}} Q_{\boldsymbol{\Phi}_n}(\boldsymbol{s}_n(t), a_n)$ in the current time slot t, user n feeds the local state $\boldsymbol{s}_n(t)$ into the DRQN, and determines the pilot sequence selection in time slot t based on the output of the DRQN.

F. Discussion

The framework in Fig. 2 is scalable as the number of DRQN modules can be varied according to the number of users without changing the subsequent training algorithm. We note that the learnable parameters of the DNNs have to be updated if the values of N or K change. However, the base station can effectively avoid frequent re-training of the DNNs by using user clustering. Assume the base station serves N' users with K' pilot sequences, but has pre-trained DNNs for N users and K pilot sequences. In this case, the base station can divide the users into $\lceil \frac{N'}{N} \rceil$ clusters, where each cluster contains N users. The base station then allocates K *orthogonal* pilot sequences to each cluster of users. This means that different clusters are allocated with different pilot sequences, and hence there is no pilot sequence collision between users in different clusters. This requires $\lceil \frac{N'}{N} \rceil K$ orthogonal pilot sequences. By doing this, the pre-trained DNNs can be re-used within each cluster of users, without invoking re-training.

We note that one user cluster may not have exactly N users when N' is not a multiple of N or N' is less than N. Moreover, one user cluster may not be allocated with exactly K pilot sequences when K' is not a multiple of K or K' is less than K. In both cases, the DNNs of the users within this particular user cluster have to be re-trained. The reason is that, as the number of users or pilot sequences in a particular cluster changes, the pilot sequence selection policies of the users within this cluster should be updated in order to adapt to the new network setting. To this end, the DNNs have to be re-trained to learn new pilot sequence selection policies.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed scheme. We simulate a GFMA system serving users with three different throughput requirements. Therefore, the users can be categorized into three groups (or types), i.e., Group I, Group II, and Group III, based on their throughput requirements. Throughout the simulations, we set the numbers of users in Group I, Group II, and Group III to be $\eta_1 N$, $\eta_2 N$, and $\eta_3 N$, respectively, where the coefficients $\eta_1, \eta_2, \eta_3 \in (0,1)$ and $\eta_1 + \eta_2 + \eta_3 = 1$. As the maximum aggregate throughput for the considered system is equal to K packets per time slot, we assume the total numbers of successful packet transmissions per time slot required by Group I, Group II and Group III users to be $\zeta_1 K$, $\zeta_2 K$, and $\zeta_3 K$, respectively, where the coefficients $\zeta_1, \zeta_2, \zeta_3 \in (0,1)$ and $\zeta_1 + \zeta_2 + \zeta_3 \leq 1$. Hence, given N and K, the throughputs required by each user in Group I, Group II, and Group III are given by $\frac{\zeta_1 K}{\eta_1 N}$, $\frac{\zeta_2 K}{\eta_2 N}$, and $\frac{\zeta_3 K}{\eta_3 N}$, respectively. We set $\frac{\zeta_1}{\eta_1} > \frac{\zeta_2}{\eta_2} > \frac{\zeta_3}{\eta_3}$ such that users in Group I have the highest throughput requirement, while Group III users have the lowest throughput requirement. The number of users is at least two times as large as the number of pilot sequences in the considered GFMA system. In each time slot, the base station trains the DNN modules for 20 iterations. The detailed parameter settings used for simulations are shown in Table I.

We compare the proposed DRL-based scheme with the following benchmark and baseline schemes[2]

[2]While a DRL-based spectrum access scheme is proposed in [21], the system model and problem setting of [21] is different from the framework we considered in this paper. Hence, we have not included the performance comparison with [21] in this paper.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Number of hidden units in LSTM D	64
Discount factor γ	0.995
Dimensions of the first and second FC layers in the DRQN	64×128 and $64 \times (K+1)$
Step size used in SGD α	0.01
Number of neurons in the hidden layer of the MLP network H	64
Number of tuples in minibatch m	80
Number of tuples in minibatch for initializing hidden states l	40
Constant in the reward function C	50
$\epsilon_{\min}$, $\epsilon_{\max}$, and ϵ_{decay} in ϵ-greedy policy	0.05, 0.95, and 2000
Coefficients η_1, η_2, and η_3 of the numbers of users in Group I, II, and III	0.25, 0.5, and 0.25
Coefficients ζ_1, ζ_2, and ζ_3 for the throughput requirements	0.35, 0.3, and 0.075

- The optimal scheme, which is obtained with the Lyapunov drift-plus-penalty algorithm as shown in the Appendix. We set $V = 1$ when evaluating the performance. With the optimal scheme, the maximum aggregate throughput can be achieved and the throughput requirements of all the users can be satisfied.
- Dynamic ACB with optimal parameter setting [25]: In this scheme, the base station first determines the number of pilot sequences allocated to Group I users, such that the pilot sequence collision probability of Group I users is minimized. The base station continues to do the same for Group II users and then allocates the remaining pilot sequences to Group III users.
- ACK-based scheme [10]: In this scheme, the base station reserves the pilot sequences that have been selected by multiple users in the previous time slot for the retransmissions of users that have suffered a packet collision. We assume no collision will happen in the retransmissions after the ACK has been sent to the users.
- Random selection scheme: Here, each user randomly selects one pilot sequence from the available ones.

A. Average Aggregate Throughput

Fig. 5 shows the evolution of the average aggregate throughput versus the time slots. We simulate $N = 12$ users sharing $K = 6$ pilot sequences. The throughput requirements of three of the users are set to $\frac{\zeta_1 K}{\eta_1 N} = 0.7$ packets per time slot (Group I users), six of the users require $\frac{\zeta_2 K}{\eta_2 N} = 0.3$ packets per time slot (Group II users), and the remaining three users require $\frac{\zeta_3 K}{\eta_3 N} = 0.15$ packets per time slot (Group III users). The proposed scheme achieves 85% of the optimal average aggregate throughput after 800 time slots, which is 31%, 128%, and 162% higher than that of the ACK-based scheme, the ACB-based scheme, and the random selection scheme, respectively. A gap exists between the aggregate throughput of the proposed scheme and the optimal scheme due to errors in the Q-value approximation and the ϵ-greedy policy in (35). In time slot 1000, we reset the system, i.e., we set the average throughput of all users back to zero. We observe that the proposed scheme quickly recovers from the new initial state after 150 time slots and retains the average aggregate throughput which is 85% of the optimum.

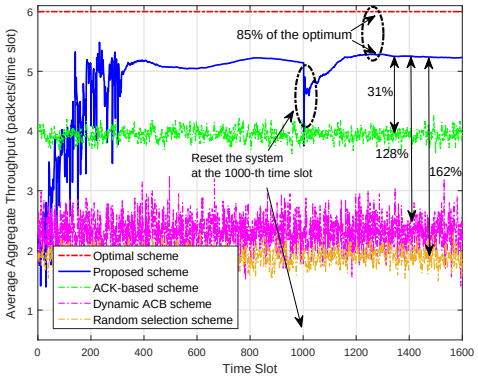
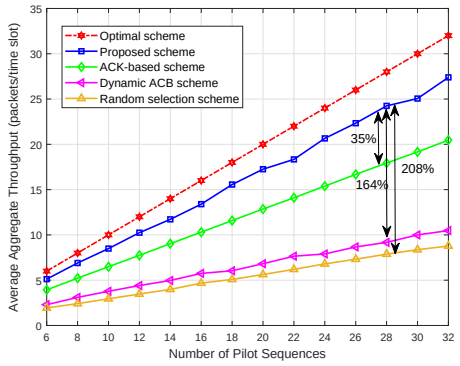
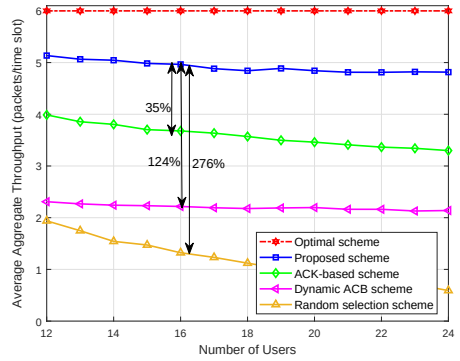


Fig. 5. Average aggregate throughput versus time slots. We set $N = 12$ and $K = 6$. The proposed scheme achieves 85% of the optimal average aggregate throughput.



(a)



(b)

Fig. 6. Average aggregate throughput versus (a) the number of pilot sequences and (b) the number of users. The performance of the proposed scheme is evaluated after the DNNs have been trained for 1500 time slots.

Fig. 6(a) 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 the DNNs have been trained for 1500 time slots. The proposed scheme can achieve an average aggregate throughput that is 35%, 164%, and 208% higher than that of the ACK-based scheme, the ACB-based scheme, and the random selection scheme, respectively, when the number of

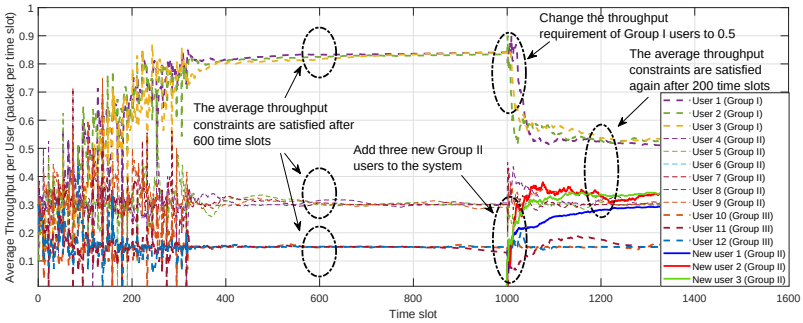


Fig. 7. Average throughput per user versus number of time slots for the proposed scheme. At the 1000-th time slot, we add three new users to Group II and change the throughput requirement of Group I users to 0.5 packets per time slot.

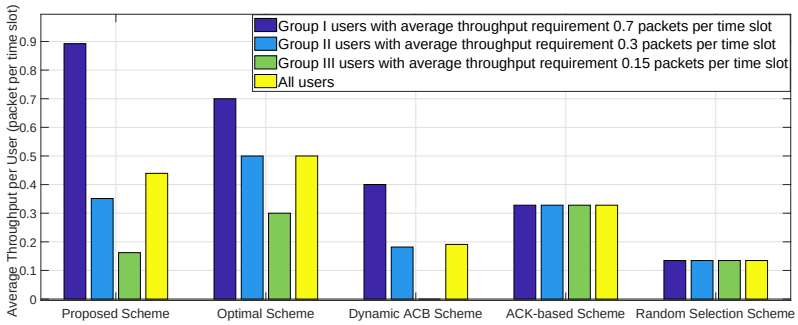


Fig. 8. Average throughput per user for users with different throughput requirements for different schemes. We set $N = 12$ and $K = 6$. The optimal solution and the proposed scheme can satisfy all average throughput requirements.

pilot sequences is 28. The near-linear increases in the average aggregate throughputs show that the probabilities of pilot sequence selection collision of all schemes do not change significantly as long as the ratio of the number of users to the number of pilot sequences is fixed. Fig. 6(b) shows the impact of the number of users on the average aggregate throughput when the number of pilot sequences is $K = 6$. The performance of the proposed scheme is evaluated after the DNNs have been trained for 1500 time slots. All schemes except for the optimal scheme suffer from a performance degradation as the number of users N increases. The reason for this is that the probabilities of pilot sequence collisions increase with N for all these schemes when the number of pilot sequences K is fixed. However, compared with the ACK-based scheme, the ACB-based scheme, and the random selection scheme, a lower collision probability can be achieved under the proposed scheme. In fact, the proposed scheme can achieve an average aggregate throughput that is 35%, 124%, and 276% higher than that of the ACK-based scheme, the ACB-based scheme, and the random selection scheme, respectively, when the number of users is 16.

B. Average Throughput of the Users

Fig. 7 shows the evolution of the achievable average throughput per user of the proposed scheme versus the time slots. We simulate $N = 12$ users sharing $K = 6$ pilot sequences. The results in Fig. 7 show that after convergence the average throughput requirements of all users are satisfied. In the proposed scheme, as all users receive the same aggregate reward, the successful transmission of a user benefits all users, while all users receive a penalty if a collision occurs or the throughput requirements are violated. Under the factorization-based MA-DRL framework, the DRQNs of the users are encouraged to learn the cooperative pilot sequence selection policies, such that a user can estimate the impact of its actions on other users from a system-level perspective. Fig. 7 also shows that joint training based on the factorization module can ensure that the policies learned during the centralized training phase can be efficiently executed in a distributed manner without having to rely on global information. Furthermore, in the 1000-th time slot, we have added three new Group II users into the system to illustrate the capability of the proposed scheme to adapt to a new network setting. We also change the throughput requirement of Group I users from 0.7 packets per time slot to 0.5 packets per time slot. We observe that, after the DNN modules have been trained for 200 time slots, the average throughputs of the three new users reach approximately 0.3 packets per time slot. Moreover, the average throughput of Group I users decrease to approximately 0.5 packets per time slot. These results show that the proposed scheme can adapt to new network topologies and new throughput requirements.

In Fig. 8, we compare the average throughput of the users for different schemes in the aforementioned setting. The performance of the proposed scheme is evaluated after the DNNs have been trained for 1500 time slots. For both the optimal scheme and the proposed scheme, all users can satisfy their average throughput requirements. For the ACB-

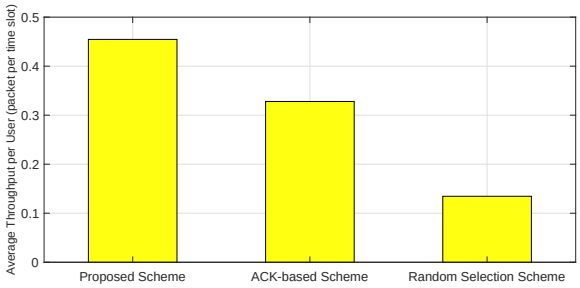


Fig. 9. Average throughput per user for different pilot sequence selection schemes for the case without user-specific throughput requirement. We set $N = 12$ and $K = 6$.

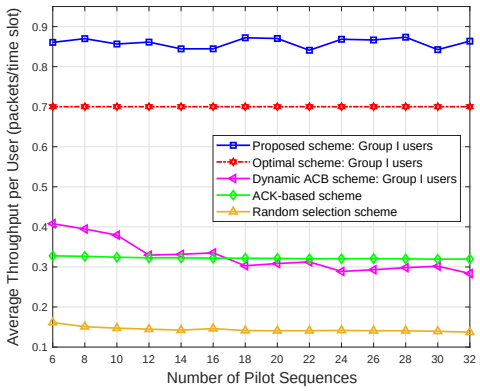


Fig. 10. Average throughput per user for different pilot sequence selection schemes versus the number of pilot sequences. We show the results of the users in Group I, while the performance of the users in Group II and Group III show a similar behavior.

based scheme, Group I users can achieve a higher average throughput than Group II users as the base station allocates more pilot sequences to Group I users. However, the users in Group I and II consume too many resources such that there is no resource left for the users in Group III to transmit. For both the ACK-based scheme and the random selection scheme, all users achieve the same average throughput as the throughput requirements are not taken into account in these two schemes.

We note that user-specific throughput requirement has not been addressed in the ACK-based scheme and the random selection scheme. Hence, for a comprehensive performance comparison, we evaluate the performance of the proposed scheme, the ACK-based scheme, and the random selection scheme for the case when there is no user-specific throughput requirement. The results for the average throughput per user of all these schemes are shown in Fig. 9. For the proposed scheme, we set coefficients $\lambda_n(t)$ in the reward function in (11) to zero. Hence, the reward function is now given by

$$R(\boldsymbol{s}(t), \boldsymbol{a}(t)) \triangleq \sum_{n \in \mathcal{N}} r_n(t), \ t \in \mathcal{T}. \tag{36}$$

The reward function corresponds to the achievable aggregate throughput (i.e., the number of successfully transmitted packets per time slot) of the system. We observe that the proposed scheme achieves the highest average throughput per user among all considered schemes. In particular, the average throughput per user of the proposed scheme is 34.6% and 222.1% higher than that of the ACK-based scheme and the random selection scheme, respectively.

Next, we investigate the average throughput of the users for different numbers of pilot sequences, and the results for the users with the highest throughput requirement, i.e., Group I users, are shown in Fig. 10. We set the number of users to $2K$. For the optimal scheme and the proposed scheme, the throughput requirements of all users are satisfied for all considered numbers of pilot sequences. Because of the centralized scheduling, for the optimal scheme, the average throughput of Group I users is exactly the same as the requirement. For the proposed scheme, Group I users tend to consume slightly more resources compared with the optimal scheme and therefore have a higher average throughput. For the ACK-based scheme, the ACB-based scheme, and the random selection scheme, the requirements of Group I users cannot be satisfied due to the relatively high probability of pilot sequence collision. We observe that, in the proposed DRL-based scheme, the DRQNs learn the near-optimal pilot sequence selection policies for a given number of pilot sequences. This also demonstrates the advantages of DRL as a model-free learning technique.

VI. Conclusion

In this paper, we proposed a DRL-based scheme for maximizing the aggregate throughput of GFMA systems, while taking into account the throughput requirements of the users. We first formulated the aggregate throughput maximization problem, where we accounted for user-specific throughput constraints. To obtain a distributed solution, we proposed an MA-DRL based scheme under a centralized training distributed execution framework. By fully exploiting global information, such as the states and actions of all the users, during the centralized training phase, the DRQNs of the users are jointly trained to learn the cooperative pilot sequence selection policies. By combining factorization with MA-DRL, the pilot sequence selection policies learned via centralized training can be efficiently executed by each user in a distributed manner. Our results showed that the proposed scheme can achieve a significant higher aggregate throughput than three previously reported schemes. The proposed scheme can also accommodate users with heterogeneous throughput requirements, and therefore has the potential to be deployed in GFMA systems supporting multiple IoT applications and services.

For future work, the extension of the proposed scheme by taking into account unsaturated traffic and the diverse quality-of-service requirements of IoT devices is an interesting topic. Furthermore, it may be possible to exploit mean-field theory [38], [39] to improve the scalability of the proposed scheme and to study the throughput optimization in large-scale GFMA systems.

Appendix

We apply stochastic network optimization to transform problem (6) into an optimization problem that can be solved

in each time slot. We use a *virtual queue* [24] to take into account the throughput requirement of user n. The dynamic of the virtual queue of user n is given by

$$q_n(t+1) \triangleq \max\left[q_n(t) - r_n(t) + \mu_n^{\text{req}}, 0\right]. \tag{A-1}$$

We define $\mathbf{q}(t) \triangleq (q_1(t), \ldots, q_N(t))$ as the vector of the backlogs of the queues of all users. We use the following Lyapunov function to measure the backlogs:

$$L(\boldsymbol{q}(t)) \triangleq \frac{1}{2}\sum_{n\in\mathcal{N}} q_n(t)^2. \tag{A-2}$$

We have the following upper bound on the conditional Lyapunov drift [24]

$$\begin{aligned}
&\Delta L(\boldsymbol{q}(t)) = \mathbb{E}[L(\boldsymbol{q}(t+1)) - L(\boldsymbol{q}(t)) \mid \boldsymbol{q}(t)] \\
&\le \mathbb{E}\left[\sum_{n\in\mathcal{N}} \frac{(\mu_n^{\text{req}})^2 + r_n(t)^2}{2} \;\middle|\; \boldsymbol{q}(t)\right] + \sum_{n\in\mathcal{N}} q_n(t)\mu_n^{\text{req}} \\
&\qquad\qquad - \mathbb{E}\left[\sum_{n\in\mathcal{N}} q_n(t) r_n(t) \;\middle|\; \boldsymbol{q}(t)\right] \\
&\le B + \sum_{n\in\mathcal{N}} q_n(t)\mu_n^{\text{req}} - \mathbb{E}\left[\sum_{n\in\mathcal{N}} q_n(t) r_n(t) \;\middle|\; \boldsymbol{q}(t)\right],
\end{aligned} \tag{A-3}$$

where B is a constant that bounds the first term on the right-hand side of the above inequality. We add $V\mathbb{E}\big[-\sum_{n\in\mathcal{N}} r_n(t) \;\big|\; \boldsymbol{q}(t)\big]$ to both sides of the inequality, where $V > 0$ is a parameter representing the importance of aggregate reward maximization. The following bound on the Lyapunov drift-plus-penalty equation can be derived

$$\begin{aligned}
&\Delta L(\boldsymbol{q}(t)) + V\mathbb{E}\left[-\sum_{n\in\mathcal{N}} r_n(t) \;\middle|\; \boldsymbol{q}(t)\right] \\
&\le B - \mathbb{E}\left[\sum_{n\in\mathcal{N}} q_n(t) r_n(t) \;\middle|\; \boldsymbol{q}(t)\right] \\
&\qquad + \sum_{n\in\mathcal{N}} q_n(t)\mu_n^{\text{req}} + V\mathbb{E}\left[-\sum_{n\in\mathcal{N}} r_n(t) \;\middle|\; \boldsymbol{q}(t)\right].
\end{aligned} \tag{A-4}$$

Given the observed $\boldsymbol{q}(t)$, as μ_n^{req} and B are constants in time slot t, minimizing the right-hand side of (A-4) can be accomplished by solving the following problem

$$\begin{aligned}
\underset{\boldsymbol{g}_n(t),\, n\in\mathcal{N}}{\text{maximize}} \quad & \sum_{n\in\mathcal{N}} \left(q_n(t) + V\right) r_n(t) \\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} g_{nk}(t) \le 1, \;\; n \in \mathcal{N}.
\end{aligned} \tag{A-5}$$

The optimal solution of problem (A-5) can be obtained in each time slot using the *Lyapunov drift-plus-penalty algorithm* [24]. In particular, the objective function of problem (A-5) can be regarded as a weighted summation of the rewards $r_n(t)$ of all users, where the weight is determined by the virtual queue of the user and the positive constant V. To obtain the optimal solution, we sort the users in descending order of their backlog, i.e., $\hat{\boldsymbol{q}}(t) \triangleq (\hat{q}_1(t), \hat{q}_2(t), \cdots, \hat{q}_N(t))$. By selecting the top K users in descending order of backlog and assigning one unique pilot sequence to each of the K users, the optimum of problem (A-5) can be obtained, which is $VK + \sum_{k=1}^{K} \hat{q}_k(t)$. Note that this requires global information and centralized scheduling.

REFERENCES

[1] R. Huang, V. W. S. Wong, and R. Schober, "Throughput optimization in grant-free NOMA with deep reinforcement learning," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Waikoloa, HI, Dec. 2019.

[2] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous Internet of Things build our future: A survey," *IEEE Commun. Surveys & Tuts.*, vol. 20, no. 3, pp. 2011–2027, Third Quarter 2018.

[3] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and Industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.

[4] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys & Tuts.*, vol. 20, no. 4, pp. 2923–2960, Fourth Quarter 2018.

[5] Cisco, "Cisco annual Internet report (2018−2023) white paper," Mar. 2020.

[6] 3GPP TS 38.213 V15.8.0, "Technical Specification Group Radio Access Network; NR; Physical layer procedures for control (Release 15)," Dec. 2019.

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

[8] J. Zhang, L. Lu, Y. Sun, Y. Chen, J. Liang, J. Liu, H. Yang, S. Xing, Y. Wu, J. Ma, I. B. F. Murias, and F. J. L. Hernando, "PoC of SCMA-based uplink grant-free transmission in UCNC for 5G," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1353–1362, Jun. 2017.

[9] 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.

[10] 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.

[11] 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.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

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

[14] D. Kalathil, N. Nayyar, and R. Jain, "Decentralized learning for multiplayer multiarmed bandits," *IEEE Trans. Inf. Theory*, vol. 60, no. 4, pp. 2331–2345, Apr. 2014.

[15] Y. Gai and B. Krishnamachari, "Distributed stochastic online learning policies for opportunistic spectrum access," *IEEE Tran. Signal Proces.*, vol. 62, no. 23, pp. 6184–6193, Dec. 2014.

[16] J. Rosenski, O. Shamir, and L. Szlak, "Multi-player bandits: A musical chairs approach," in *Proc. of Int'l Conf. Machine Learning (ICML)*, Jun. 2016.

[17] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Sydney, Australia, 2017.

[18] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. of Int'l Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, Stockholm, Sweden, Jul. 2018.

[19] T. Rashid, M. Samvelyan, C. S. Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Stockholm, Sweden, Jul. 2018.

[20] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 4, no. 2, pp. 257–265, Jun. 2018.

[21] 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.

[22] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1277–1290, Jun. 2019.

[23] J. R. Kok and N. Vlassis, "Sparse cooperative Q-learning," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Banff, Canada, Jul. 2004.

[24] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[25] S. Duan, V. Shah-Mansouri, Z. Wang, and V. W. S. Wong, "D-ACB: Adaptive congestion control algorithm for bursty M2M traffic in LTE networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9847–9861, Dec. 2016.

[26] J. Ahn, B. Shim, and K. B. Lee, "EP-based joint active user detection and channel estimation for massive machine-type communications," *IEEE Trans. Commun.*, vol. 67, no. 7, pp. 5178–5189, Jul. 2019.

[27] 3GPP TS 38.214 V16.1.0, "Technical specification group radio access network; NR; Physical layer procedures for data (Release 16)," Apr. 2020.

[28] 3GPP TS 23.501 V16.4.0, "Technical specification group services and system aspects; System architecture for the 5G system (5GS); Stage 2 (Release 16)," Mar. 2019.

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

[30] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proc. of Int'l Conf. on Machine Learning (ICML)*, Williamstown, MA, Jun. 2001.

[31] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.

[32] 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.

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

[34] K. Tumer and D. H. Wolpert, *Collectives and the Design of Complex Systems*. Springer, 2004.

[35] G. Weiss, *Multiagent Systems*, 2nd ed. MIT Press, 2013.

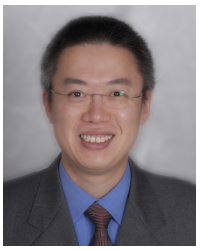
[36] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," *arXiv preprint arXiv:1803.00933*, 2018.

[37] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating functional knowledge in neural networks," *J. Mach. Learn. Res.*, vol. 10, pp. 1239–1262, Jun. 2009.

[38] H. Kim, J. Park, M. Bennis, S. Kim, and M. Debbah, "Mean-field game theoretic edge caching in ultra-dense networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 935–947, Jan. 2020.

[39] B. Zhou and W. Saad, "Age of information in ultra-dense IoT systems: Performance and mean-field game analysis," *arXiv preprint arXiv:2006.15756*, Jun. 2020.

Rui Huang (S'19) received the B.Eng. degree from Chongqing University, Chongqing, China, in 2015, and the M.Eng. degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 2018. He is currently a Ph.D. Candidate in the Department of Electrical and Computer Engineering, The University of British Columbia (UBC), Vancouver, Canada. He has been a recipient of the Four Year Doctoral Fellowship (4YF) at UBC since 2018. His research interests include Internet of Things (IoT) and machine learning for optimization in wireless communication systems.



Vincent W.S. Wong (S'94, M'00, SM'07, F'16) received the B.Sc. degree from the University of Manitoba, Winnipeg, MB, Canada, in 1994, the M.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada, in 1996, and the Ph.D. degree from the University of British Columbia (UBC), Vancouver, BC, Canada, in 2000. From 2000 to 2001, he worked as a systems engineer at PMC-Sierra Inc. (now Microchip Technology Inc.). He joined the Department of Electrical and Computer Engineering at UBC in 2002 and is currently a Professor. His research areas include protocol design, optimization, and resource management of communication networks, with applications to wireless networks, smart grid, mobile edge computing, and Internet of Things. Currently, Dr. Wong is an Executive Editorial Committee Member of *IEEE Transactions on Wireless Communications*, an Area Editor of *IEEE Transactions on Communications* and *IEEE Open Journal of the Communications Society*, and an Associate Editor of *IEEE Transactions on Mobile Computing*. He is a Technical Program Co-chair of the *IEEE* 92*nd Vehicular Technology Conference (VTC*2020*-Fall)*. He has served as a Guest Editor of *IEEE Journal on Selected Areas in Communications* and *IEEE Wireless Communications*. He has also served on the editorial boards of *IEEE Transactions on Vehicular Technology* and *Journal of Communications and Networks*. He was a Tutorial Co-Chair of *IEEE Globecom*'18, a Technical Program Co-chair of *IEEE SmartGridComm*'14, as well as a Symposium Co-chair of *IEEE ICC*'18, *IEEE SmartGridComm* ('13, '17) and *IEEE Globecom*'13. He is the Chair of the IEEE Vancouver Joint Communications Chapter and has served as the Chair of the IEEE Communications Society Emerging Technical Sub-Committee on Smart Grid Communications. He is an IEEE Communications Society Distinguished Lecturer (2019 - 2020).

Robert Schober (S'98, M'01, SM'08, F'10) received the Diplom (Univ.) and the Ph.D. degrees in electrical engineering from Friedrich-Alexander University of Erlangen-Nuremberg (FAU), Germany, in 1997 and 2000, respectively. From 2002 to 2011, he was a Professor and Canada Research Chair at the University of British Columbia (UBC), Vancouver, Canada. Since January 2012 he is an Alexander von Humboldt Professor and the Chair for Digital Communication at FAU. His research interests fall into the broad areas of Communication Theory, Wireless Communications, and Statistical Signal Processing.

Robert received several awards for his work including the 2002 Heinz Maier Leibnitz Award of the German Science Foundation (DFG), the 2004 Innovations Award of the Vodafone Foundation for Research in Mobile Communications, a 2006 UBC Killam Research Prize, a 2007 Wilhelm Friedrich Bessel Research Award of the Alexander von Humboldt Foundation, the 2008 Charles McDowell Award for Excellence in Research from UBC, a 2011 Alexander von Humboldt Professorship, a 2012 NSERC E.W.R. Stacie Fellowship, and a 2017 Wireless Communications Recognition Award by the IEEE Wireless Communications Technical Committee. Since 2017, he has been listed as a Highly Cited Researcher by the Web of Science. Robert is a Fellow of the Canadian Academy of Engineering and a Fellow of the Engineering Institute of Canada. From 2012 to 2015, he served as Editor-in-Chief of the *IEEE Transactions on Communications*. Currently, he serves as Member of the Editorial Board of the *Proceedings of the IEEE* and as VP Publications for the IEEE Communication Society (ComSoc).