Partial Computation Offloading in NOMA-Assisted Mobile Edge Computing Systems Using Deep Reinforcement Learning

Van Dat Tuong, Thanh Phung Truong, The-Vi Nguyen, Wonjong Noh, and Sungrae Cho

***Abstract*—Mobile edge computing (MEC) and non-orthogonal multiple access (NOMA) have been regarded as promising technologies for beyond fifth-generation (B5G) and sixth-generation (6G) networks. This study aims to reduce the computational overhead (weighted sum of consumed energy and latency) in a NOMA-assisted MEC network by jointly optimizing the computation offloading policy and channel resource allocation under dynamic network environments with time-varying channels. To this end, we propose a deep reinforcement learning algorithm named ACDQN that utilizes the advantages of both actor-critic and deep Q-network methods and provides low complexity. The proposed algorithm considers partial computation offloading, where users can split computation tasks so that some are performed on the local terminal while some are offloaded to the MEC server. It also considers a hybrid multiple access scheme that combines the advantages of NOMA and orthogonal multiple access (OMA) to serve diverse user requirements. Through extensive simulations, it is shown that the proposed algorithm stably converges to its optimal value, provides approximately 10%, 27%, and 69% lower computational overhead than the prevalent schemes such as full offloading with NOMA, random offloading with NOMA, and fully local execution, and achieves near-optimal performance.**

***Index Terms*—Deep reinforcement learning, mobile edge computing (MEC), non-orthogonal multiple access (NOMA), partial computation offloading, resource allocation**

I. Introduction

THE explosive growth in mobile Internet services has resulted in the development of a variety of computation-hungry applications, e.g., augmented/virtual reality, three-dimensional (3D) gaming, online artificial intelligence, smart factory, and big data analytics for Internet of Things (IoTs), which impose a heavy computational burden on mobile terminals/users with limited computation and power resources [1].

To address this issue, mobile edge computing (MEC) has been introduced as a promising solution [2], [3]. The key idea of MEC is to provide resourceful computing capability to servers located at the edge of radio access networks, e.g., base stations (BSs), so that mobile users can offload their computation workloads to the edge computing servers. Many studies have demonstrated the effectiveness of MEC in completing computationally intensive tasks [4]–[11]. However, although MEC can fulfill the high computational requirements of users, offloading tasks to the MEC server results in a certain amount of delay and incurs energy. Therefore, in order to better utilize the advantages of MEC, some methods are needed to reduce the delay and energy consumption.

Non-orthogonal multiple access (NOMA) is considered as one of the promising candidates for meeting this demand. By allowing users to simultaneously transmit data over the same resource block (RB) and using successive interference cancellation (SIC) to decode individual signals among users, NOMA is capable of accommodating more users than orthogonal multiple access (OMA) technique [12]. Although NOMA schemes are more complex as they involve superimposing and decoding of wireless signals, due to the advantages of NOMA, many recent studies have been devoted to investigating the challenges involved in NOMA transmission [13]–[15].

Therefore, in a NOMA-assisted MEC system, which utilizes the advantages of both MEC and NOMA, multiple mobile users can offload their tasks more efficiently using the same resource. For example, let us assume that only one RB is unoccupied at a given moment and that two users will offload their tasks to the MEC server. If OMA transmission is applied, only one user can offload, while the other must wait. However, if NOMA is applied, both users can offload to the MEC server simultaneously, which can reduce the latency and energy consumption when the decoding energy consumption is negligible [16]. Hence, the computing services in B5G and 6G can benefit from using a NOMA-assisted MEC system, which is a very important communication technique in future wireless networks and has received considerable attention recently.

The conventional optimization approach involves heavy computations, which make it challenging to apply it in a dynamic NOMA network where the solution is required to be in real-time. Recent advances in artificial intelligence, e.g., deep learning (DL), can address this problem. By optimizing a model over all possible state realizations, DL algorithms can significantly reduce the complexity of determining solutions at different times. To the best of our knowledge, there is no machine-learning based work that has studied the joint optimization problem of partial computation offloading and resource allocation to minimize the energy consumption and latency in a hybrid NOMA-MEC network. The main contributions and difference of this study can be summarized as follows:

Manuscript received .....; revised .....; and accepted February 28 2021. This work was supported by the Korea Electric Power Corporation under Grant R19XO01-41 and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2020R1F1A106911911). *(Corresponding author: Wonjong Noh and Sungrae Cho.)*

V. D. Tuong, T. P. Truong, T.-V. Nguyen and S. Cho are with the School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea. *(e-mail: vdtuong@uclab.re.kr, tptruong@uclab.re.kr, tvnguyen@uclab.re.kr, srcho@cau.ac.kr)*

W. Noh is with the School of Software, Hallym University, Chuncheon 24252, Republic of Korea. *(e-mail: wonjong.noh@hallym.ac.kr)*

- We formulate a joint optimization problem of computation offloading policy and channel resource allocation as a non-convex mixed integer programming (MIP) model. It aims to determine the optimal computation offloading amount and NOMA-OMA subcarrier allocation considering channel conditions. As a solution, we propose a low-complexity deep reinforcement learning (DRL)-based algorithm, which can avoid the complexity inherent in recomputing solutions at different times by directly utilizing the trained model of deep neural networks (DNNs).
- Many studies have focused on binary offloading in NOMA-assisted MEC, while only a few researches have investigated the partial computation offloading [8]–[11]. However, it is more suitable to offload tasks partially rather than fully for efficient utilization of the limited bandwidth in wireless networks [17]. Also, many studies have focused on the resource allocation in conventional pure NOMA-assisted MEC, while only a few researches have investigated MEC with hybrid NOMA and OMA protocols [13]–[15]. Unlike the methods in prior studies [8]–[11], [13]–[15], the algorithm proposed in our study finds the joint optimal control that enables partial computation offloading and hybrid multiple access controls under practical constraints such as SIC complexity and maximum power constraints for the NOMA channel.
- The proposed algorithm combines the advantages of both actor-critic and deep Q-network (DQN) methods, which is jointly referred to as ACDQN. In the algorithm, we employ primary and target DNNs to stably train the computation offloading policy whose domain is continuous between 0 and 1. In addition, we employ ϵ-greedy strategy to select the channel resource allocation that can be determined from indicators, e.g., 0 or 1. Then, the comprehensive joint action is aggregated in critic DNNs and it is evaluated using DQN method with Q-values and instant rewards.
- We analyze the complexity of the proposed algorithm, which has polynomial time and space complexity. Numerical results are presented to demonstrate the reliable convergence and its near-optimal performance, by the comparison with NOMA-OMA based prevalent offloading schemes and exhaustive search, in reducing energy consumption and latency for task completion of all users.

The rest of the paper is organized as follows. Section II introduces the related studies. In Section III, the system model and problem formulation are described. Section IV presents our proposed algorithm based on the DRL algorithm. The performance evaluation is discussed in Section V. Finally, in Section VI, we conclude the paper and outline our future work.

II. Related Work

Recently, many studies have proposed ways to tackle the technical challenges of NOMA-assisted MEC systems. Among them, some studies focused on energy minimization [18]–[24]. Kiani *et al.* [18] proposed and demonstrated the benefits of a MEC-aware NOMA approach for 5G networks by jointly optimizing user clustering, computation and communication resource allocation, and transmission powers to minimize the energy consumption of mobile users. Wu *et al.* [19] investigated an energy-efficient multitask computation offloading scheme in NOMA MEC networks. They exploited the varying delay limits of tasks and the differing computation rates of edge nodes to minimize the total energy consumption. Ye *et al.* [20] studied an energy-efficient communication scheme from a system perspective for distributed NOMA MEC networks. They formulated a joint optimization problem for the computing frequencies of the MEC server and mobile users, offloading time, transmission power, and local and remote task-execution time, which was then solved using a Dinkelbach-based algorithm. Eliodorou *et al.* [21] tried to minimize the overall energy consumption for all users by jointly optimizing user association, optimal power allocation, data rate, and offloaded data. In particular, two coalition game algorithms were proposed and compared in an effort to efficiently reduce the total energy consumption. In [22], the energy consumption minimization problem was formulated with joint optimization of time assignment, power control, CPU frequency, and computation offloading. By exploiting the block coordinate descent (BCD) method, it developed a joint communication and computation resource allocation algorithm to address the original non-convex problem. Specifically, the optimal solution was obtained in closed form. In [23], both power and time allocations were jointly optimized to reduce the energy consumption of computation offloading. Closed-form expressions for the optimal power and time allocations were obtained and used to establish the conditions for determining whether the conventional OMA, pure NOMA, or hybrid NOMA should be used for MEC offloading. Li *et al.* [24] tried to minimize the energy consumption of a hybrid NOMA-assisted MEC system. The original energy minimization problem is non-convex, and to solve it, a multilevel programming method was proposed. This method decomposes the non-convex problem into three subproblems, namely power allocation, time slot scheduling, and offloading task assignment, which are solved optimally by carefully studying their convexity and monotonicity. In addition, a close-to-optimal algorithm with low complexity was proposed.

Compared to the energy minimization problems, minimizing the delay is more challenging, since delay is a ratio between rate-related functions. Some studies have focused on latency minimization [25]–[29]. In [25] and [26], the authors studied optimization problems of computation workloads and the time required for uploading workloads as well as downloading computed results with NOMA to minimize the overall workload-completion delay. Sheng *et al.* [27] proposed another approach to reduce the computation delay by characterizing the interaction between differentiated uploading delay and co-channel interference, and then iteratively determining NOMA user pairing and offloading policy using semidefinite relaxation and convex-concave procedure. Wu *et al.* [28] tried to minimize the overall delay in performing computation tasks, by jointly optimizing the offloaded workloads and the NOMA transmission time. Despite the non-convexity of the formulated joint optimization problem, they proposed distributed efficient algorithms to find the optimal offloading solution.

Ding *et al.* [29] tried to minimize the offloading delay for NOMA-assisted MEC systems. By transforming the delay minimization problem into a form of fractional programming, two iterative algorithms based on Dinkelbach's method and Newton's method were proposed. Furthermore, a criterion for choosing between three possible modes, namely OMA, pure NOMA, and hybrid NOMA, for MEC was established.

Some studies have focused on joint minimization of energy consumption and latency [30], [31]. Zhu *et al.* [30] considered jointly optimizing the power and time allocations to reduce the delay and energy consumption in hybrid NOMA and MEC. The main contribution of this study was the characterization of the optimal power and time allocations in a closed form. In addition, by incorporating a matching algorithm with the optimal power and time allocations, it proposed a low-complexity method to efficiently optimize user grouping. Pham *et al.* [31] proposed utilizing cooperative game theory to solve the computation offloading problem in multicarrier NOMA MEC networks to minimize the overall computational overhead in terms of energy consumption and latency. It was demonstrated that this solution ensures convergence to a Nash equilibrium.

Specifically, Ye *et al.* [32] focused on successful computation probability. They proposed a new hybrid offloading scheme in a NOMA MEC network that can operate in three different modes, namely partial offloading, complete local computation, and complete offloading. It provides closed-form mathematical expressions of the successful computation probability and its optimal solutions for these three schemes.

Several studies have proposed DL-based solutions for NOMA MEC networks [33]–[38]. Yang *et al.* [33] considered a NOMA-MEC system with multiple users and a single MEC server, and investigated the problem of minimizing latency. By using the DQN algorithm to select users who offload at the same time without knowing the actions of other users in advance, it obtains the optimal user combination state and minimizes the system offloading latency. The authors in [34] and [35] extended the schemes in [19] and [31], respectively. In particular, they developed DRL-based algorithms for reducing the total energy consumption of IoT devices with a latency limit [34] and increasing the computation rate of the MEC server [35]. Meanwhile, NOMA-MEC systems are vulnerable to various attacks such as denial of service attacks and rogue edge attacks. Xiao *et al.* [36], [37] investigated different attack models in MEC systems, focusing on both mobile offloading and caching procedures. In [36], RL-based security solutions were proposed to provide secure offloading to the edge nodes against jamming attacks. It also presented light-weight authentication and secure collaborative caching schemes to protect data privacy. In [37], an RL-based mobile offloading scheme for edge computing was proposed to prevent jamming attacks and interference. It uses safe RL to avoid choosing the risky offloading policy that fails to meet the computational latency requirements of the tasks. Doan *et al.* [38] developed a method to maximize the probability that all users decode the desired signals in order to optimize the quality of service (QoS) while ensuring fairness among users. This was achieved by finding an RL-based power control in cache-aided NOMA networks.

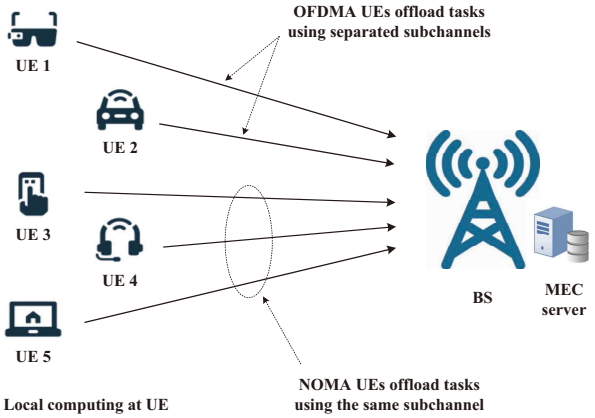


Fig. 1. Mobile devices partially offload tasks to the MEC server using hybrid NOMA–OMA subchannels.

III. System Model and Problem Formulation

A. Network Model

As illustrated in Fig. 1, we consider a network model consisting of U user equipments (UEs) and one MEC server co-located with a BS. We assume that time is slotted, and the time slot and time slot index set are denoted by t and $\mathcal{T} \triangleq \{0, 1, \ldots, \}$, respectively. The wireless channel is assumed to be independent identically distributed (i.i.d.) block fading, i.e., the channel remains static within each time slot, but varies among different time slots.

Let $\mathcal{U} = \{u_i |\ i = 1, \ldots, U\}$ and $\mathcal{S} = \{s_i |\ i = 1, \ldots, S\}$ denote the sets of UEs and orthogonal SCs, respectively. In this work, we assume that the number of UEs is significantly greater than the number of SCs, i.e., $U \gg S$. In addition, some UEs share NOMA SCs while other UEs use OMA SCs. This type of implementation can be viewed as a hybrid NOMA–OMA scheme. Assuming that there are M OMA SCs, e.g., orthogonal frequency division multiple access (OFDMA) SCs and N NOMA SCs, M and N can take any value in the range $[0, S]$ that satisfies $M + N = S$. Without loss of generality, we use $\mathcal{M} = \{s_i |\ i = 1, \ldots, M\}$ and $\mathcal{N} = \{s_i |\ i = M+1, \ldots, S\}$ to denote the sets of OMA and NOMA SCs, respectively; then, we have $\mathcal{M} \cup \mathcal{N} = \mathcal{S}$.

We assume that each OFDMA SC is occupied by at most one UE; hence, it is clear that M OFDMA SCs will be utilized by at most M UEs. On the other hand, we use $\mathcal{U}_{s_i}$ ($s_i \in \mathcal{N}$) to denote the set of UEs sharing the NOMA SC s_i. By utilizing NOMA SCs, the received signal of a UE contains not only its desired signal but also the interfering signals of the co-sharing UEs. The SIC is applied at the receiving ends to decode individual signals from the aggregated received signal. However, a NOMA SC may be overloaded with too many UEs that are active on it, which results in an infeasible SIC complexity. In [39], the authors proved that SIC complexity could be reduced significantly by setting a small number of active UEs per NOMA SC. Similarly, in this study, we set an upper limit on the number of active UEs per NOMA SC, which is denoted by L. Therefore, we have $|\mathcal{U}_{s_i}| \leq L, \forall s_i \in \mathcal{N}$, and the total number of concurrent active UEs does not exceed a fixed number, e.g., $(M + LN)$.

There are many applications that may profit from computation offloading [2]. In general, they can be classified into three major groups: data partition oriented applications, code partition oriented applications, and continuous execution applications [8]. In this study, we mainly focus on data partition oriented applications such as data compression, online gaming, and applications based on augmented/virtual reality, which require high computational capability and low execution delay to enhance user experience. In addition, their computation tasks can be partitioned for parallel execution in a local device and MEC server.

B. Communication Model

We denote $\mathbf{x}(t) = \{x_{u_i,s_j}(t) |\ u_i \in \mathcal{U},\ s_j \in \mathcal{S}\}$ as the channel allocation, where $x_{u_i,s_j}(t) \in \{0,1\}$ indicates the association between UE u_i and BS via SC s_j. Here, s_j can be any NOMA or OMA SC. We have $x_{u_i,s_j}(t) = 1$ if at time t, UE u_i is associated with BS via SC s_j, otherwise $x_{u_i,s_j}(t) = 0$. Let $h_{u_i,s_j}(t)$ denote the channel gain between UE u_i and BS via SC s_j, which includes distance-dependent loss, shadowing loss, antenna gain, and instantaneous fading. Without loss of generality, we assume that the UEs sharing the NOMA SC s_i ($s_i \in \mathcal{N}$) are ordered as $h_{u_j,s_i}(t) \leq \ldots \leq h_{u_k,s_i}(t)\ \forall u_j, u_k \in \mathcal{U}_{s_i} | j < k$. According to [40], the SIC decoding order in the uplink (UL) follows the descending order of channel gain. This means that the UL signal of NOMA UE with the greatest channel gain will be decoded and removed from the aggregated signal first, considering the UL signals of other co-sharing UEs as interference. Afterward, the UL signals of other co-sharing UEs will be decoded step by step in accordance with descending channel gain order. By adopting Shannon formula, we can express the UL rate of UE u_i on NOMA SC s_j ($s_j \in \mathcal{N}$) as

$$R_{u_i,s_j}(t) = B \log_2 \left(1 + \frac{x_{u_i,s_j}(t) p_{u_i} h_{u_i,s_j}(t)}{I_{u_i,s_j}(t) + \sigma^2}\right), \tag{1}$$

where B is the SC bandwidth, p_{u_i} is the transmission power of UE u_i, σ^2 is the additive white Gaussian noise power spectral density, and $I_{u_i,s_j}(t)$ is the aggregated interference from the co-sharing UEs impacting UE u_i, which can be computed as

$$I_{u_i,s_j}(t) = \sum_{u_k \in \mathcal{U}_{s_j} | k<i} x_{u_k,s_j}(t) p_{u_k} h_{u_k,s_j}(t). \tag{2}$$

Moreover, because all SCs are orthogonal, the intra-cell interference among OFDMA UEs as well as between an OFDMA UE and a NOMA UE is completely eliminated. Therefore, the UL rate of UE u_i on OFDMA SC s_j ($s_j \in \mathcal{M}$) can be expressed as

$$R_{u_i,s_j}(t) = B \log_2 \left(1 + \frac{x_{u_i,s_j}(t) p_{u_i} h_{u_i,s_j}(t)}{\sigma^2}\right). \tag{3}$$

In this work, we assume that the transmission power of all UEs is predefined, and to enhance the QoS, the UL transmission power of co-sharing UEs should be subject to the total transmission power constraint [40]. Let P_{tot} denote the allowed total transmission power on a NOMA SC. Accordingly, the total transmission power constraint of UEs sharing NOMA SC s_j ($s_j \in \mathcal{N}$) can be expressed as

$$\sum_{u_i \in \mathcal{U}_{s_j}} x_{u_i,s_j}(t) p_{u_i} \leq P_{tot}. \tag{4}$$

C. Computation Model

We denote the computation offloading policy by $\mathbf{o}(t) = \{o_{u_i}(t) |\ u_i \in \mathcal{U}\}$, where $o_{u_i}(t) \in [0,1]$ specifies the ratio of the computation task of UE u_i, which is offloaded to the MEC server. The remaining ratio, $(1 - o_{u_i}(t))$, is executed locally. Note that we have two special cases: $o_{u_i}(t) = 1$ indicates that UE u_i offloads its computation task entirely to the MEC server, whereas $o_{u_i}(t) = 0$ indicates that the computation task is executed locally without any offloading. We define the computation task of UE u_i as $W_{u_i} = \{Q_{u_i}, C_{u_i}\}$, where Q_{u_i} is the task size in bits, and C_{u_i} is the number of CPU cycles required to accomplish W_{u_i}. In this study, we aim to minimize the total computational overhead in terms of energy consumption and latency for completing the tasks of all power-hungry UEs instead of the MEC server.

1) Local Computing: We denote $f^l_{u_i}$ as the computing capability of UE u_i in *CPU-cycles/second*. The time required to execute a computation task partially at UE u_i is computed as

$$T^l_{u_i}(t) = (1 - o_{u_i}(t)) \frac{C_{u_i}}{f^l_{u_i}}. \tag{5}$$

By utilizing the widely adopted model of energy consumption per computing cycle as $\mathcal{E} = \kappa f^2$ [41], [42], where κ is an energy coefficient that depends on the hardware architecture and f is the CPU frequency, the energy consumed while executing the computation task partially at UE u_i can be computed as

$$E^l_{u_i}(t) = (1 - o_{u_i}(t)) Q_{u_i} \kappa_{u_i} (f^l_{u_i})^2, \tag{6}$$

where κ_{u_i} is the energy coefficient of UE u_i.

2) Computation Offloading: In the case of computation offloading, the time required for completing the computation task of UE u_i is primarily composed of two parts: UL transmission delay $T^b_{u_i}(t)$ and remote-execution delay $T^c_{u_i}(t)$. In this study, we assume that the size of the computed results is small so that time required for superimposing and downloading the computed results from the MEC server is negligible. Moreover, because the SIC complexity is significantly reduced due to the upper limit on the number of co-sharing NOMA UEs, we do not take into account the signal decoding delay. Therefore, the time required for completing the computation task of UE u_i during computation offloading can be computed as

$$\begin{aligned} T^r_{u_i}(t) &= T^b_{u_i}(t) + T^c_{u_i}(t) \\ &= \frac{o_{u_i}(t) Q_{u_i}}{R_{u_i,s_j}(t)} + \frac{o_{u_i}(t) C_{u_i}}{f^r_{u_i}} \\ &= o_{u_i}(t) \left(\frac{Q_{u_i}}{R_{u_i,s_j}(t)} + \frac{C_{u_i}}{f^r_{u_i}}\right), \end{aligned} \tag{7}$$

where $f^r_{u_i}$ is the amount of MEC computation resources allocated to UE u_i, which is assumed to be predefined at each

time slot. We compute the energy consumption $E^r_{u_i}(t)$ of user u_i for uploading its computation task to the MEC server as

$$E^r_{u_i}(t) = \frac{p_{u_i}}{\zeta_{u_i}} T^b_{u_i}(t) = \frac{p_{u_i}}{\zeta_{u_i}} \frac{o_{u_i}(t) Q_{u_i}}{R_{u_i,s_j}(t)}, \tag{8}$$

where ζ_{u_i} is the power amplifier efficiency of UE u_i.

3) Total Computational Overhead: In the partial offloading scheme, the task-completion latency $T_{u_i}(t)$ of UE u_i accounts for the longest time delay among the delays $T^l_{u_i}(t)$ of local computing and $T^r_{u_i}(t)$ of computation offloading, and it is given by

$$\begin{aligned} T_{u_i}(t) &= \max\left(T^l_{u_i}(t), T^r_{u_i}(t)\right) \\ &= \max\left(\left(1 - o_{u_i}(t)\right)\frac{C_{u_i}}{f^l_{u_i}}, o_{u_i}(t)\left(\frac{Q_{u_i}}{R_{u_i,s_j}(t)} + \frac{C_{u_i}}{f^r_{u_i}}\right)\right), \end{aligned} \tag{9}$$

Unlike the latency, the energy consumption $E_{u_i}(t)$ of UE u_i is the sum of $E^l_{u_i}(t)$ in local computing and $E^r_{u_i}(t)$ in computation offloading, which is computed as

$$\begin{aligned} E_{u_i}(t) &= E^l_{u_i}(t) + E^r_{u_i}(t) \\ &= \left(1 - o_{u_i}(t)\right) Q_{u_i} \kappa_{u_i} \left(f^l_{u_i}\right)^2 + \frac{p_{u_i}}{\zeta_{u_i}} \frac{o_{u_i}(t) Q_{u_i}}{R_{u_i,s_j}(t)}. \end{aligned} \tag{10}$$

Furthermore, the total computational overhead $Z_{u_i}(t)$ of UE u_i can be determined in terms of the energy consumption $E_{u_i}(t)$ and latency $T_{u_i}(t)$. Similar to [7], we define $Z_{u_i}(t)$ as the weighted sum of energy consumption and latency as follows:

$$Z_{u_i}(t) = \beta^e_{u_i} E_{u_i}(t) + \beta^l_{u_i} T_{u_i}(t), \tag{11}$$

where the weight parameters $\beta^e_{u_i}$ and $\beta^l_{u_i}$ specify the UE preferences regarding energy consumption and latency, respectively, in which $\beta^e_{u_i}, \beta^l_{u_i} \in [0, 1]$ and $\beta^e_{u_i} + \beta^l_{u_i} = 1$, $\forall u \in \mathcal{U}$.

D. Problem Formulation

In this study, we minimize the total computational overhead by jointly optimizing the channel resource allocation $\mathbf{x}(t)$ and the computation offloading policy $\mathbf{o}(t)$. This optimization problem can be formulated as

$$\min_{\mathbf{x}(t),\mathbf{o}(t)} \quad Z(\mathbf{x}(t), \mathbf{o}(t)), \tag{12}$$

$$\text{s.t.} \quad x_{u_i,s_j}(t) \in \{0, 1\}, \forall u_i \in \mathcal{U}, \forall s_j \in \mathcal{S}, \tag{13}$$

$$\sum_{s_j \in \mathcal{S}} x_{u_i,s_j}(t) \leq 1, \forall u_i \in \mathcal{U}, \tag{14}$$

$$o_{u_i}(t) \in [0, 1], \forall u_i \in \mathcal{U}, \tag{15}$$

$$o_{u_i}(t) \leq \max(\{x_{u_i,s_j}(t) | s_j \in \mathcal{S}\}), \forall u_i \in \mathcal{U}, \tag{16}$$

$$|\mathcal{U}_{s_j}| \leq L, \forall s_j \in \mathcal{N}, \text{ with } L \ll U, \tag{17}$$

$$\sum_{u_i \in \mathcal{U}_{s_j}} x_{u_i,s_j}(t) p_{u_i} \leq P_{tot}, \forall s_j \in \mathcal{N}. \tag{18}$$

In (12), we denote the total computational overhead incurred by all UEs by $Z(\mathbf{x}(t), \mathbf{o}(t))$, which can be expressed as

$$\begin{aligned} Z(\mathbf{x}(t), \mathbf{o}(t)) &= \sum_{u_i \in \mathcal{U}} Z_{u_i}(t) \\ &= \sum_{u_i \in \mathcal{U}} \beta^e_{u_i} E_{u_i}(t) + \beta^l_{u_i} T_{u_i}(t). \end{aligned} \tag{19}$$

Constraints (13) and (14) describe the channel resource allocation, i.e., each UE can utilize at most one SC. Constraint (15) specifies the computation offloading policy by which computation tasks can be processed in parallel using both local and remote executions, and constraint (16) states that only associated users can offload their computation tasks. In addition, constraints (17) and (18) ensure that the number of co-sharing UEs and the allowed total power over one NOMA SC do not exceed the predefined values L and P_{tot}, respectively. These conditions are necessary to enhance the QoS with efficient and low-complexity SIC.

The problem formulated in (12) is an MIP problem because of the existence of multiple discrete and continuous variables. Moreover, MIP problems are known to be NP-hard by nature, and finding the optimal solution usually requires exponential time complexity [43]. As the channels between UEs and BS change dynamically over time, a large number of possible channel realizations can be generated at different times, which makes it challenging to apply conventional optimization solutions in real-time. To address this issue, we study and develop a novel DL solution based on DRL algorithm.

IV. Deep Reinforcement Learning for Partial Computation Offloading in NOMA MEC Systems

A. RL Task Formulation

We consider the BS as an RL agent and the entire network system as the environment in RL jargon. At each time slot, the agent interacts with the environment by deciding an action based on the collective information regarding channel conditions. The action includes channel resource allocation and computation offloading policy for all UEs. After executing the action, the agent achieves a step reward, which is computed from the computational overhead defined in (19). In addition, it is updated with an evolved environmental state. We identify the state space, action space, and reward function as follows:

1) State space: The environmental state is determined by realizing the channel conditions between UEs and BS on all SCs. Therefore, the state $s(t)$ at time t can be formulated based on the realistic channel gains as follows:

$$s(t) = \begin{bmatrix} h_{u_1,s_1}(t), & \ldots, & h_{u_1,s_S}(t), \\ h_{u_2,s_1}(t), & \ldots, & h_{u_2,s_S}(t), \\ \ldots, & \ldots, & \ldots, \\ h_{u_U,s_1}(t), & \ldots, & h_{u_U,s_S}(t) \end{bmatrix}. \tag{20}$$

2) Action space: At time t, the agent decides an aggregated action $a(t)$ including the channel resource allocation $\mathbf{x}(t)$ and the computation offloading policy $\mathbf{o}(t)$. Therefore, the action $a(t)$ can be given as follows:

$$a(t) = \begin{bmatrix} x_{u_1,s_1}(t), & \ldots, & x_{u_1,s_S}(t), \\ x_{u_2,s_1}(t), & \ldots, & x_{u_2,s_S}(t), \\ \ldots, & \ldots, & \ldots, \\ x_{u_U,s_1}(t), & \ldots, & x_{u_U,s_S}(t), \\ o_{u_1}(t), & \ldots, & o_{u_U}(t) \end{bmatrix}, \tag{21}$$

which must satisfy all constraints (13)–(18) of the optimization problem given in (12). The action yields a significantly low reward if any constraint is violated.

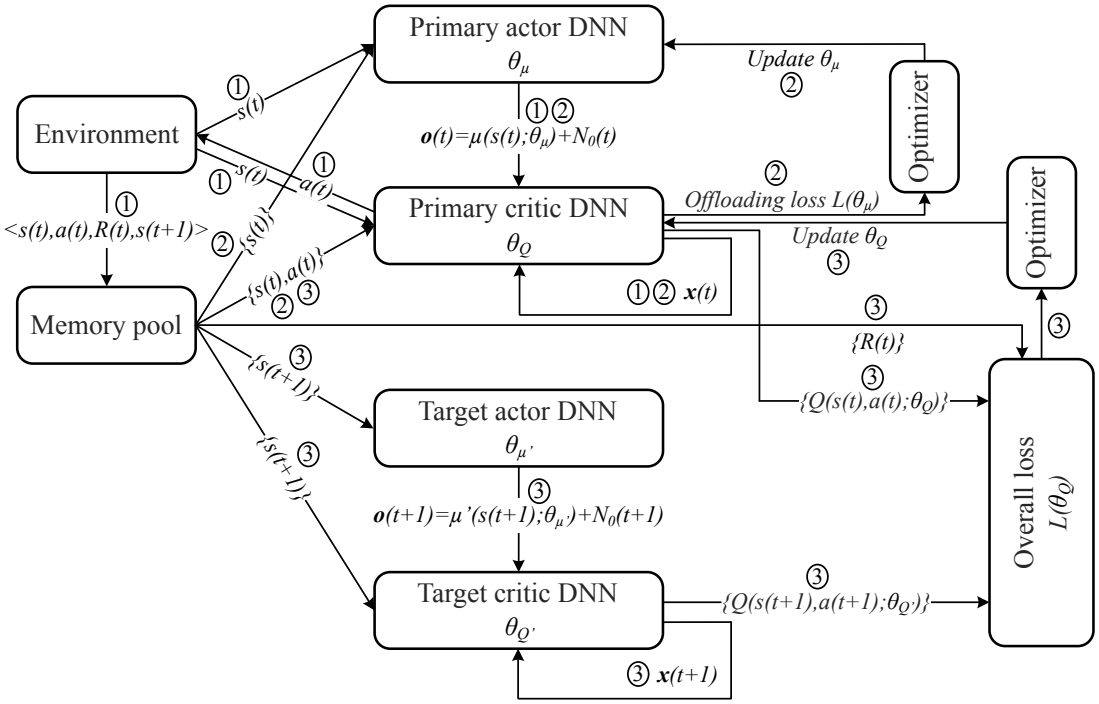


Fig. 2. Structure of the proposed ACDQN algorithm: (1) collecting experiences through interaction with the environment, (2) training the optimal computation offloading policy, and (3) training the optimal overall action including channel resource allocation and computation offloading policy.

3) Reward function: In NOMA MEC systems, the computational overhead in terms of the energy consumption and latency is a crucial metric for evaluating the system performance. We define a reward function $R(t)$ as an inversion of the computational overhead as follows:

$$\begin{aligned} R(t) &= -\ Z(\mathbf{x}(t), \mathbf{o}(t)) \\ &= -\sum_{u_i \in \mathcal{U}} \left(\beta^e_{u_i} E_{u_i}(t) + \beta^l_{u_i} T_{u_i}(t) \right). \end{aligned} \tag{22}$$

At time t, the agent observes a state $s(t)$, selects and performs an action $a(t)$, and then obtains a step reward $R(t)$ and the next state $s(t+1)$. It aims to determine the optimal action that can maximize the long-term return R^{long}, which is computed from the step rewards with discounting factor γ. Therefore, we can model the optimization problem (12) as an RL task as follows:

$$R^{long} = \max_{a(t)} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t R(t) \right] \tag{23}$$

where γ^t approaches zero when $t \to T-1$ is large enough.

B. ACDQN Algorithm

Motivated by the advantages of DRL-based algorithms [44]–[46] in handling discrete and continuous action spaces, we propose and develop a novel DRL algorithm for addressing the RL task (23) as well as the optimization problem (12).

1) Overall Architecture: The architecture of the algorithm is illustrated in Fig. 2 that comprises three processes: (1) collecting experiences through interaction with the environment, (2) training the optimal computation offloading policy, and (3) training the optimal overall action including channel resource allocation and computation offloading policy. They can be described as follows.

The first process involves the environment, a memory pool, and the primary actor and critic DNNs. The agent observes a state $s(t)$ from the environment and feeds it to the primary actor and critic DNNs. The continuous computation offloading policy $\mathbf{o}(t)$ is selected in the primary actor DNN μ using weight θ_μ as follows:

$$\mathbf{o}(t) = \mu(s(t); \theta_\mu) + N_0(t), \tag{24}$$

where $N_o(t)$ is the noise that is added to enhance the exploration of the computation offloading policy. In practice, we can use Gaussian white noise or Ornstein–Uhlenbeck (OU) noise [46], which is a correlated additive Gaussian noise generated by $dN_0(t) = \vartheta(\upsilon - N_0(t))dt + \sigma dW(t)$, where ϑ is a drift coefficient, υ is the mean value, σ is a diffusion coefficient, and $W(t)$ is the standard Wiener process. The discrete channel resource allocation $\mathbf{x}(t)$ is selected in the primary critic DNN using weight θ_Q and ϵ-greedy strategy as

$$x_{u_i,s_j}(t) = \begin{cases} w & \text{if } z \le \epsilon \\ \text{argmax}_{x_{u_i,s_j}(t)} Q(s(t), a(t); \theta_Q) & o.w., \end{cases} \tag{25}$$

where w is a random variable generating 0 or 1 equiprobably, $z \in [0,1]$ is a random variable, and $\epsilon \in [0,1]$ is a monotone decreasing random variable, which first generates a number close to 1 and gradually decreases the number to 0 along with the training process. The agent interacts with the environment by executing the overall action $a(t) = \mathbf{x}(t) \oplus \mathbf{o}(t)$, where $\oplus$ denotes an aggregation operation, which results in a step reward $R(t)$ and the next state $s(t+1)$. The experience tuple $\langle s(t), a(t), R(t), s(t+1) \rangle$ is then saved to the memory pool.

The second process involves the memory pool, the primary actor and critic DNNs, and an optimizer, e.g., Adam optimizer [47]. From the memory pool, the agent samples a D-size minibatch of experiences to feed into the DNNs, where $D \in (1, F)$ with F is the size of the memory pool. Similar to the first process, the continuous computation offloading policy $\mathbf{o}(t)$ is selected based on (24), while the discrete channel resource allocation $\mathbf{x}(t)$ is fixed in this process. In addition, because the computation offloading policy $\mathbf{o}(t)$ is continuous, the Q-value function can be assumed to be differentiable with respect to $\mathbf{o}(t)$. Therefore, we can construct a gradient to update weight θ_μ of the primary actor DNN μ as follows:

$$\begin{aligned}\nabla_{\theta_\mu} L(\theta_\mu) &= \mathbb{E}\left[\nabla_{\theta_\mu} Q(s(t), a(t))\right] \\ &= \mathbb{E}\left[\nabla_{\mathbf{o}(t)} Q(s(t), a(t)) \nabla_{\theta_\mu} \mathbf{o}(t)\right] \\ &= \mathbb{E}\left[\nabla_{\mathbf{o}(t)} Q(s(t), a(t)) \nabla_{\theta_\mu} \left(\mu(s(t); \theta_\mu) + N_0(t)\right)\right],\end{aligned} \tag{26}$$

where $L(\theta_\mu)$ is the Q-value loss based on the computation offloading policy, the so-called offloading loss.

The third process involves the memory pool, the primary critic DNN, the target actor and critic DNNs, and an optimizer, e.g., Adam optimizer [47]. Basically, this process operates like a DQN algorithm, in which the primary and target critic DNNs are utilized for computing the Q-value loss $L(\theta_Q)$ based on the overall action, the so-called overall loss. From the memory pool, the agent samples a D-size minibatch of experiences to feed into the primary critic DNN and the target actor and critic DNNs. By handling the input of state-action pairs, the primary critic DNN delivers the Q-value $Q(s(t), a(t); \theta_Q)$ at the current state using its weight θ_Q. Similar to the primary actor and critic DNNs in the first process, the target actor and critic DNNs select the continuous computation offloading policy $\mathbf{o}(t+1)$ and the discrete channel resource allocation $\mathbf{x}(t+1)$, respectively. After that, the overall action for next state $s(t+1)$ is aggregated as $a(t+1) = \mathbf{x}(t+1) \oplus \mathbf{o}(t+1)$. The target critic DNN delivers the Q-value $Q(s(t+1), a(t+1); \theta_{Q'})$ at the next state using its weight $\theta_{Q'}$. Therefore, the overall loss can be given as

$$\begin{aligned}L(\theta_Q) = \frac{1}{D} \sum_{i=1}^{D} \Bigg(& R_i(t) + \gamma Q'\left(s_i(t+1), a_i(t+1); \theta_{Q'}\right) \\ & - Q(s_i(t), a_i(t); \theta_Q) \Bigg)^2,\end{aligned} \tag{27}$$

where γ is the discounting factor. The optimizer is utilized to update the weight θ_Q of the primary critic DNN for minimizing the overall loss. The weights of the target actor and critic DNNs can be soft-updated based on the weights of the primary actor and critic DNNs as follows:

$$\theta_{\mu'} = \tau \theta_\mu + (1 - \tau) \theta_{\mu'}, \tag{28}$$

$$\theta_{Q'} = \tau \theta_Q + (1 - \tau) \theta_{Q'}, \tag{29}$$

where τ is the common target learning rate (LR) for updating the target actor and critic weights.

Algorithm 1 DRL algorithm for partial computation offloading in NOMA MEC systems (ACDQN)

% Initialization
1: Initialize the network model.
2: Initialize DNN weights θ_μ and θ_Q of the primary actor μ and primary critic Q, respectively.
3: Initialize DNN weights $\theta\mu' \leftarrow \theta_\mu$ and $\theta_{Q'} \leftarrow \theta_Q$ of the target actor μ' and target critic Q', respectively.
4: Initialize experience replay memory $\mathcal{F}$ with size F.
% Training
5: **while** the stop condition is not satisfied **do**
6: Get initial state $s(1)$ based on (20).
7: **for** t = 1, ..., T **do**
8: Get state $s(t)$.
9: Select overall action $a(t)$ including $\mathbf{o}(t)$ and $\mathbf{x}(t)$.
10: Execute action $a(t)$.
11: Compute reward $R(t)$ and next state $s(t+1)$.
12: Save experience $\langle s(t), a(t), R(t), s(t+1) \rangle$ to $\mathcal{F}$.
13: Sample experiences to feed into DNNs.
14: Compute offloading and overall losses.
15: Update weight θ_μ using Adam optimizer [47].
16: Update weight θ_Q using Adam optimizer [47].
17: **if** after every G steps **then**
18: Soft-update weights $\theta_{\mu'}$ and $\theta_{Q'}$.
19: **end if**
20: **end for**
21: **end while**
22: Return $\theta_\mu^* = \theta_\mu$ and $\theta_Q^* = \theta_Q$.

2) Training Algorithm: The training algorithm is described in Algorithm 1. It first initializes the network model, weights of the primary actor and critic DNNs, weights of the target actor and critic DNNs, and the experience replay memory (see lines 1–4). In each episode, the agent obtains the initial state by collecting information about the channel conditions between UEs and BS (see line 6). In each training step, it gets the current state, and based on this state, it selects to perform an action including the computation offloading policy based on (24) and the channel resource allocation based on (25) (see lines 8–10). Afterward, a step reward is computed based on (22), and the environment evolves to the next state (see line 11). Next, the experience tuple $\langle s(t), a(t), R(t), s(t+1) \rangle$ is saved to the replay memory (see line 12). The weights of the primary actor and critic DNNs are updated using Adam optimizer [47] to minimize the offloading and overall losses (see lines 13–16). The weights of the target actor and critic DNNs are soft-updated after every G steps based on (28) and (29), respectively (see lines 17–19). The training process ends once it meets the stop condition, e.g., it reaches to the maximum number of episodes or the overall loss becomes less than a preset value. Finally, it returns the optimal weights θ_μ^* and θ_Q^* of the primary actor and critic DNNs for later exploitation (see line 22). For example, by exploiting the returned weights θ_μ^* and θ_Q^*, we can determine the overall action $a(t)$ for a state $s(t)$, i.e., $a(t) = \mathbf{x}(t) \oplus \mathbf{o}(t)$, where $\mathbf{o}(t) = \mu(s(t); \theta_\mu^*)$ and $\mathbf{x}(t)$ is given as $x_{u_i, s_j}(t) = \text{argmax}_{x_{u_i, s_j}(t)} Q(s(t), a(t); \theta_Q^*)$.

TABLE I
SYSTEM-LEVEL SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
Network coverage radius, $\mathcal{R}$	100 m	SC bandwidth, B	1 MHz
Number of UEs, U	10–15	Number of SCs, S, (N/M)	4–8 (2/2, 3/2, 3/3, 4/3, 4/4)
Min. dist. between UEs and the BS	5 m	Fading model	Rayleigh
Total number of co-sharing UEs, L	1–4	BS antenna pattern	0 dB (omnidirectional)
UE transmission power, p_{u_i}	20–23 dBm	Shadowing deviation	10 dB
Total power of co-sharing UEs, P_{tot}	26 dBm	Noise power, σ^2	−175 dBm/Hz
Energy switched coefficient, κ_{u_i}	1e-28 [5]	Path loss	$140.7 + 36.7\log_{10}(d(\text{km}))$ [dB]
MEC allocated computing resources, $f^r_{u_i}$	4–10 GHz	Energy/Latency weight, $\beta^l_{u_i}/\beta^e_{u_i}$	0.5/0.5 [31]
UE computing capability, $f^l_{u_i}$	1 GHz	Task size, Q_{u_i}	420–900 KB [9]
Number of training episodes, E	2,000	Task complexity, C_{u_i}	1.68–3.6 GHz
Number of steps per episode, T	300	Primary learning rates, $\text{LR}_\mu/\text{LR}_Q$	1e-3/1e-3, 1e-4/1e-3, 1e-4/1e-4
Replay memory size, F	250,000	Common target learning rate, τ	1e-3
Discounting factor, γ	0.9	OU noise coefficient, (υ, ϑ, σ)	(0, 0.15, 0.2)
Minibatch size, D	32	OU noise decaying factor, DF	1e-4, 1e-5, 1e-6, 0

C. Computational Complexity

We derive the space and time complexity of the proposed algorithm based on the replay memory and the DNN workload. The replay memory's size is fixed. When the number of experiences becomes so large that the memory capacity becomes insufficient, a number of oldest experiences will be freed to allocate memory for the new incoming experiences. Regarding the neural network architecture, both the primary actor and critic DNNs have one input layer, two hidden layers, and one output layer, which are all fully connected. We use the rectified linear unit (ReLU) function as the activation function for the hidden layers. The primary actor DNN μ uses the tangent (tanh) function to deliver the output computation offloading policy. In the primary critic DNN, the output Q-values are obtained linearly. Let g_1, g_2 and g'_1, g'_2 denote the corresponding numbers of neurons in the hidden layers of the primary and critic DNNs, respectively. Then, according to [48], we can compute the space complexity of the primary actor and critic DNNs, respectively, as

$$O^{space}_{\mu} = O((US)^2) + O(g_1^2 + g_2^2) + O(U^2), \tag{30}$$

$$O^{space}_{Q} = O((US)^2) + O(g'^2_1 + g'^2_2) + O((US+U)^2). \tag{31}$$

Similar to [49], the upper bound of the time complexity of the primary actor and critic DNNs can be given, respectively, as

$$O^{time}_{\mu} = O(ET((US)^2 + g_1^2 + g_2^2 + U^2)), \tag{32}$$

$$O^{time}_{Q} = O(ET((US)^2 + g'^2_1 + g'^2_2 + (US+U)^2)), \tag{33}$$

where E is the number of training episodes and T is the maximum number of training steps in an episode. Because the target actor and critic DNNs share the same neural network structures with the corresponding primary actor and critic DNNs, their space and time complexity are identical, e.g., $O^{space/time}_{\mu} = O^{space/time}_{\mu'}$, and $O^{space/time}_{Q} = O^{space/time}_{Q'}$. Therefore, the space and time complexity of the proposed algorithm can be computed, respectively, as

$$\begin{aligned} O^{space} = {} & O(2((US)^2 + g_1^2 + g_2^2 + U^2)) \\ & + O(2((US)^2 + g'^2_1 + g'^2_2 + (US+U)^2)), \end{aligned} \tag{34}$$

$$\begin{aligned} O^{time} = {} & O(2ET((US)^2 + g_1^2 + g_2^2 + U^2)) \\ & + O(2ET((US)^2 + g'^2_1 + g'^2_2 + (US+U)^2)). \end{aligned} \tag{35}$$

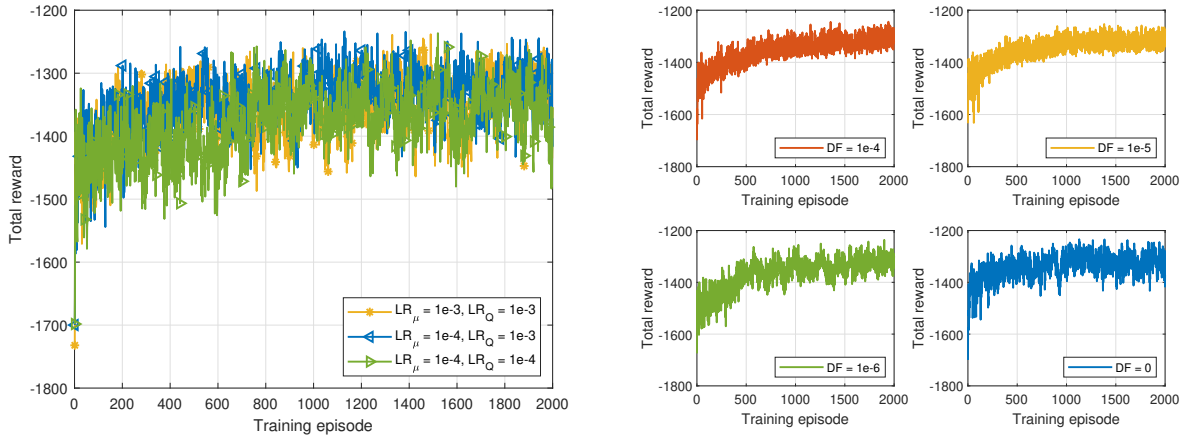
We see that the complexity of the proposed algorithm is polynomial and suitable for application to practical problems.

On the other hand, in the hybrid NOMA–OMA scheme, each channel resource allocation is a permutation that involves choosing $(M+LN)$ UEs out of the total U UEs. Therefore, the complexity of exhaustive search (ES) in the channel resource allocation problem can be given as $O(P_U^{M+LN})$ with

$$P_U^{M+LN} := \binom{U}{M+LN} = \frac{U!}{(U-(M+LN))!}. \tag{36}$$

For the partial computation offloading problem, the computation offloading policy is a continuous variable in the range $[0, 1]$ so that the complexity of ES is $O(\infty)$. Assuming that a quantization to Ω levels ensures the existence of a near optimal computation offloading solution, the complexity of ES is reduced to $O(\Omega^{M+LN})$. Accordingly, the overall complexity of ES in the joint optimization problem of partial computation offloading and channel resource allocation is given as $O(P_U^{M+LN}\Omega^{M+LN})$. Therefore, we can see that the complexity of the proposed algorithm is much lower than that of the ES algorithm.

Remark 1. In the exploitation phase, we utilize the weights θ^*_μ and θ^*_Q returned from Algorithm 1 to instantaneously determine the optimal computation offloading policy $\mathbf{o}^*(t)$ and the optimal channel resource allocation $\mathbf{x}^*(t)$ for any state $s(t)$, i.e., $\mathbf{o}^*(t) = \mu(s(t); \theta^*_\mu)$ and $\mathbf{x}^*(t)$ is given as $x_{u_i,s_j}(t) = \text{argmax}_{x_{u_i,s_j}(t)} Q(s(t), a(t); \theta^*_Q)$. Therefore, comparing with the conventional optimization approach, the proposed algorithm significantly reduces the complexity.



(a) Impact of primary LRs

(b) Impact of the OU noise DF

Fig. 3. Convergence of the proposed algorithm with various primary DNN learning rates (LRs) and OU noise decaying factors (DFs): $U = 10$, $S = 5$, $N = 3$, $M = 2$, $L = 2$, $f_{u_i}^l = 1$ GHz, $f_{u_i}^r = 4$ GHz, $Q_{u_i} = 420$ KB, $C_{u_i} = 1.68$ GHz, $E = 2000$, and $T = 300$.

V. Performance Evaluation

A. Simulation Settings

We built the network model and developed the proposed algorithm using PyTorch with Python 3.7.3 on a server powered by Intel Core i5-8500 CPU with Nvidia GTX 1050 Ti GPU, and 250 GB of memory. The network model comprised an MEC server attached to a BS, i.e., an access point in a warehouse or a roadside unit, having a coverage radius of 100 m. The UEs were uniformly distributed within the coverage area and were placed apart from the BS by at least 5 m. The transmission power of the UEs was equally selected from the set $\{20, 21, 22, 23\}$ dBm, and the total transmission power of co-sharing NOMA UEs did not exceed 26 dBm. The baseline number of SCs was five, including three NOMAs and two OFDMAs, each having a bandwidth of 1 MHz. We utilized a path loss model of $140.7 + 36.7\log_{10}(\text{d(km)})$ and an i.i.d. Rayleigh block fading model with a shadowing deviation of 10 dB. We employed fully connected actor and critic DNNs, in which the number of neurons in their two corresponding hidden layers were 128–128 and 1024–512, respectively. We considered an application of augmented reality framework ARkit [9] for all UEs, in which the ranges of computation task size and required CPU cycles were adjusted to $[420, 900]$ KB and $[1.68, 3.6]$ GHz, respectively. In addition, we assumed that the tasks were generated regularly at every time step and equally across all UEs. The system level simulation parameters are summarized in Table I. To evaluate the performance of the proposed algorithm, we compared it with several existing schemes. Details are as follows:

- *Proposed scheme (ACDQN algorithm):* The computation tasks of the associated UEs are optimally partitioned for local execution and offloading to the MEC server using NOMA and OMA SCs (ACDQN-NOMA), and OMA SCs only (ACDQN-OMA).
- *Random offloading (RAO):* The computation tasks of the associated UEs are randomly partitioned for local execution and offloading to the MEC server using NOMA and OMA SCs.
- *Fully remote computing (FRC):* All the computation tasks of the associated UEs are fully offloaded to the MEC server using NOMA and OMA SCs.
- *Fully local computing (FLC):* All the computation tasks of UEs are executed locally.
- *Deep deterministic policy gradient (DDPG):* This scheme utilizes DDPG algorithm as a regular DRL method for continuous control.
- *Near optimum with exhaustive search (ES):* This scheme searches whole combinations of offloading and resource allocation. Here, the continuous computation offloading policy is quantized equally into a deterministic level. Therefore, it is considered a near-optimal scheme.

B. Convergence

Fig. 3 (a) and (b) depict the convergence of the proposed algorithm with various primary DNN learning rates (LRs) and OU noise decaying factors (DFs). Here, the primary DNN LRs LR_μ and LR_Q are used by the Adam optimizer for updating the weights θ_μ and θ_Q of the primary actor and critic DNNs, respectively. In addition, we use DF to reduce the OU noise perturbation after each training step by using the control of $\vartheta(t) = \vartheta(t) \cdot \max(0, 1 - \text{DF} \cdot t)$ in the OU noise coefficients. The baseline network setting is as follows: $U = 10$, $S = 5$, $M = 3$, $N = 2$, $L = 2$, $f_{u_i}^l = 1$ GHz, $f_{u_i}^r = 4$ GHz, $Q_{u_i} = 420$ KB, and $C_{u_i} = 1.68$ GHz. The number of training episodes and training steps per episode are $E = 2000$ and $T = 300$, respectively.

From Fig. 3 (a), we can see that the total reward increases as the number of training episodes increases, and it converges within a specific range. This is because the raw channel states are inputted directly for training without any quantization. In addition, the highest total reward achieved with $(\text{LR}_\mu, \text{LR}_Q) = (1e{-}4, 1e{-}3)$ is slightly higher than those with $(\text{LR}_\mu, \text{LR}_Q) = (1e-3, 1e-3)$ and $(\text{LR}_\mu, \text{LR}_Q) = (1e-4, 1e-4)$. Therefore,

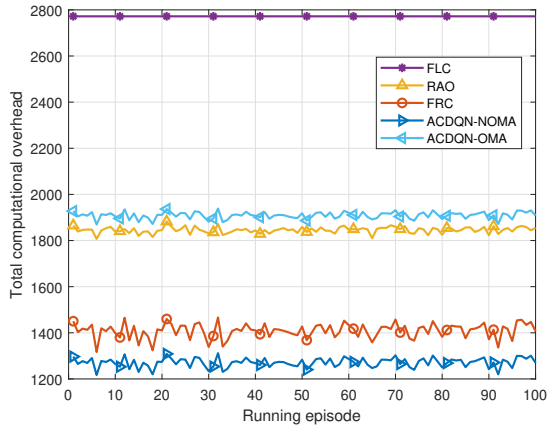


Fig. 4. Total computational overhead comparison between the proposed algorithm and existing schemes: $U = 10$, $S = 5$, $N = 3$, $M = 2$, $L = 2$, $f^l_{u_i} = 1$ GHz, $f^r_{u_i} = 4$ GHz, $Q_{u_i} = 420$ KB, and $C_{u_i} = 1.68$ GHz.

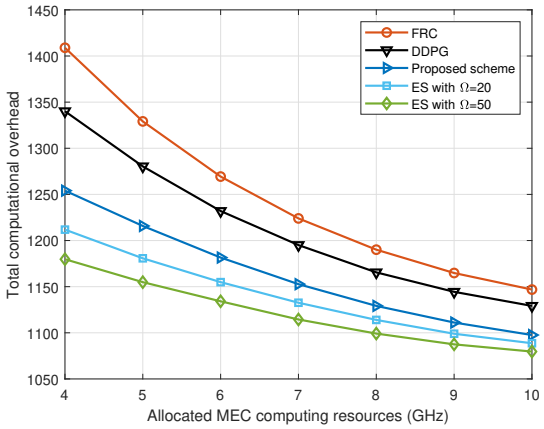


Fig. 5. Comparison of the total computational overhead based on the allocated MEC computing resources: $U = 10$, $S = 5$, $N = 3$, $M = 2$, and $L = 2$.

we select the training result with $(\mathrm{LR}_\mu, \mathrm{LR}_Q) = (1e{-}4, 1e{-}3)$ for evaluating the performance of the proposed algorithm.

In Fig. 3 (b), we can see that the proposed algorithm converges in all DF settings such as DF = $1e-4$, DF = $1e-5$, DF = $1e-6$, and DF = 0. In particular, the total reward with smaller DFs, i.e., DF = $1e-6$ and DF = 0, fluctuates more than that with larger DFs, i.e., DF = $1e-4$ and DF = $1e-5$. This is because smaller DFs provide more opportunities to explore the continuous action space. In practice, we prefer small DFs as it provides a greater chance of achieving the highest reward.

C. Performance Comparisons

We save the weights θ^*_μ and θ^*_Q of the primary actor and critic DNNs that provide the highest total reward for determining the optimal computation offloading policy and channel resource allocation. For evaluating the performance of the proposed scheme, we compare it with other schemes through 100 consecutive episodes, each including 300 time steps. Fig. 4 plots the fluctuation of the total computational overhead. From the figure, we can see that the total computational overhead with ACDQN-NOMA is approximately 10%, 46%, and 118% less than those with FRC, RAO, and FLC, respectively. In particular, compared to the ACDQN-OMA scheme, the total computational overhead is reduced significantly by 51%, which demonstrates the effectiveness of NOMA over OMA in MEC systems.

Fig. 5 shows the impact of the allocated MEC computing resources on the total computational overhead. For this evaluation, we vary $f^r_{u_i}$ from 4 to 10 GHz. As the allocated MEC computing resources increase, the total computational overhead decreases slightly in all offloading schemes such as FRC, DDPG algorithm, proposed scheme with ACDQN algorithm, and ES algorithm. Specifically, the amount of reduction becomes smaller. This is because a larger allocation of MEC computing resources reduces only the remote execution delay, while the energy consumption and delay for uploading tasks to the MEC server remain unchanged and they significantly contribute to the total computational overhead.

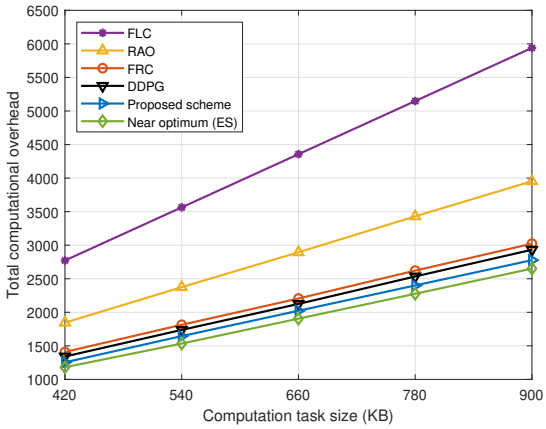


Fig. 6. Comparison of the total computational overhead with respect to the size of the computation tasks: $U = 10$, $S = 5$, $N = 3$, $M = 2$, and $L = 2$.

Furthermore, we can see that the proposed scheme outperforms FRC and DDPG schemes. In particular, the total computational overhead in the proposed scheme is very close to that achieved with ES algorithm. For instance, when $f^r_{u_i} = 10$ GHz, the proposed scheme provides a total computational overhead that is only 1.5% and 2.2% greater than those achieved with ES algorithm when $\Omega = 20$ and $\Omega = 50$, respectively.

In Fig. 6, we observe the impact of the computation task size on the total computational overhead. For this evaluation, we vary Q_{u_i} from 420 to 900 KB. As the computation task size increases, the total computational overhead increases in all schemes. In particular, the increasing amount in FLC scheme is significantly larger than those in other schemes. This is because unlike other schemes, in FLC, all the UEs have to execute their tasks without any assistance from the MEC server. Furthermore, we can see that the total computational overhead in the proposed scheme is less than those provided by FRC, RAO, FLC, and DDPG algorithm. Specifically, it is very close to that achieved with ES algorithm. For instance, when $Q_{u_i} = 900$ KB, the total computational overhead with ACDQN algorithm is 2769, which is approximately 10%, 43%, 115%,

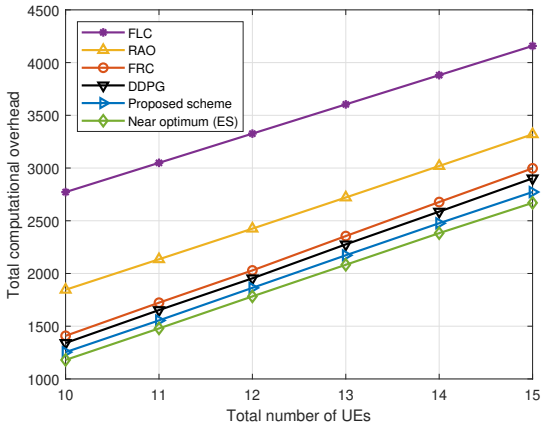


Fig. 7. Comparison of the total computational overhead with respect to the number of UEs: $S = 5$, $N = 3$, $M = 2$ and $L = 2$.

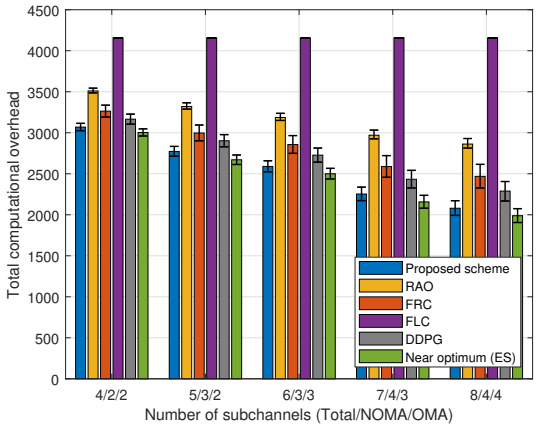


Fig. 8. Comparison of the total computational overhead with respect to the number of SCs including NOMAs and OFDMAs: $U = 15$ and $L = 2$.

and 6% less than those provided by FRC, RAO, FLC, and DDPG algorithm, respectively; and it is only 4% greater than the near optimum provided by ES algorithm with $\Omega = 50$.

In Fig. 7, we investigate the total computational overhead by varying the number of UEs. At each time slot, the channel resource is optimally allocated to a limited number of UEs. Accordingly, as the total number of UEs increases, the number of UEs that have to execute their tasks without MEC assistance increases. This causes a regular increase in the total computational overhead. In particular, as the number of UEs increases from 10 to 15, the total computational overhead increases regularly in all schemes, e.g., it increases from 2772 to 4158 in FLC scheme, from 1846 to 3320 in RAO scheme, from 1409 to 2997 in RFC scheme, from 1341 to 2902 in DDPG scheme, from 1254 to 2773 in the proposed scheme with ACDQN algorithm, and from 1181 to 2698 in the near optimal scheme with ES algorithm ($\Omega = 50$).

Fig. 8 shows the performance comparison by varying the number of SCs including NOMAs and OFDMAs. We set the number of UEs to 15 and vary the number of SCs from 4 to 8 with a corresponding increase in the number of NOMA and

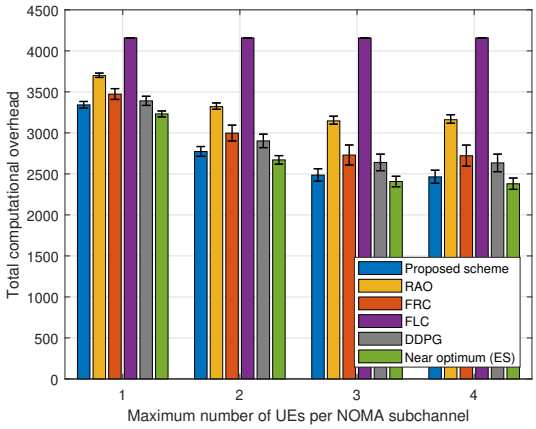


Fig. 9. Comparison of the total computational overhead with respect to the maximum number of UEs per NOMA SC: $U = 15$, $S = 5$, $N = 3$, $M = 2$, $P_{\text{tot}} = 26$ dBm.

OFDMA SCs from 2 to 4. In the x-axis, the number of SCs S, NOMAs N, and OFDMAs M are denoted by $(S/N/M)$. We can see that as the number of SCs increases, the total computational overhead decreases in offloading schemes such as the proposed scheme, RAO, FRC, DDPG algorithm, and the near optimal scheme with ES algorithm ($\Omega = 50$). In addition, the performance gaps between the proposed scheme and the existing schemes (RAO, FRC, and FLC) become larger. For example, when $(S/N/M) = (4/2/2)$, the total computational overhead in the proposed scheme is 3119, which is approximately 5%, 13%, and 33% less than those with FRC, RAO, and FLC, respectively, and when $(S/N/M) = (8/4/4)$, these gaps increase to 14%, 38%, and 100%, respectively. Furthermore, we see that the proposed scheme with ACDQN algorithm outperforms DDPG algorithm as its performance is close to the near optimum provided by ES algorithm.

Fig. 9 shows the performance comparison with respect to the number of co-sharing UEs per NOMA SC. When $L = 1$, it is the same as the scheme employing all OFDMA SCs and no NOMA SC. When L is increased from 1 to 3, the total computational overhead significantly decreases in offloading schemes such as the proposed scheme, RAO, FRC, DDPG algorithm, and the near optimal scheme with ES algorithm ($\Omega = 50$). However, the computational overhead reduces slightly when L increases from 3 to 4. This is because the total transmission power of the co-sharing NOMA UEs is limited to $P_{\text{tot}} = 26$ dBm (398.1 mW) so that up to three UEs with transmission powers of 20 dBm (100 mW) or 21 dBm (125.9 mW) are allowed to share one NOMA SC. From the figure, we can see that the proposed scheme outperforms the existing schemes (RAO, FRC, and FLC) and DDPG algorithm. In addition, the performance of the proposed scheme is close to the near optimum achieved by ES algorithm.

VI. Conclusion

In this study, we investigated the joint optimization problem of partial computation offloading and channel resource allocation in NOMA-assisted MEC systems to minimize the overall

computational overhead in terms of energy consumption and latency. Here, we assumed a dynamic network environment with time-varying channels, which requires fast and energy-efficient determination of optimal solutions for different states. To this end, we proposed a low-complexity DRL algorithm that uses both actor-critic and DQN methods. We first found a real-time optimal computation offloading policy, in which UEs partition computation tasks into multiple subtasks for simultaneous execution at the UE and MEC server. Second, we found a real-time optimal channel resource allocation in the hybrid NOMA–OMA system, in which the network provides NOMA and OMA SCs to concurrently serve various users with diverse requirements. Extensive numerical results confirmed that the proposed ACDQN algorithm reliably converges. It also outperforms the existing offloading schemes using OMA and NOMA, and achieves near-optimal performance in terms of energy consumption and latency-based computational overhead. As our future work, we will expand this study to multiserver systems with massive MIMO, in which users located at the boundary between multiple MEC servers jointly select the best server, resource, and antenna precoding to offload their tasks for saving energy and reducing latency.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1628–1656, 2017.

[3] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1657–1681, 2017.

[4] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, 2017.

[5] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, 2017.

[6] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, 2018.

[7] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, 2018.

[8] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, 2016.

[9] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, 2018.

[10] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, 2019.

[11] M. Sheng, Y. Wang, X. Wang, and J. Li, "Energy-Efficient Multiuser Partial Computation Offloading With Collaboration of Terminals, Radio Access Network, and Edge Server," *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1524–1537, 2019.

[12] P. Wang, J. Xiao, and P. Li, "Comparison of orthogonal and non-orthogonal approaches to future wireless cellular systems," *IEEE Veh. Technol. Mag.*, vol. 1, no. 3, pp. 4–11, 2006.

[13] N. Nomikos, T. Charalambous, D. Vouyioukas, G. K. Karagiannidis, and R. Wichman, "Hybrid NOMA/OMA with buffer-aided relay selection in cooperative networks," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 524–537, 2019.

[14] X. Shao, C. Yang, D. Chen, N. Zhao, and F. R. Yu, "Dynamic IoT device clustering and energy management with hybrid NOMA systems," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4622–4630, 2018.

[15] M. Zeng, A. Yadav, O. Dobre, and H. V. Poor, "Energy-efficient joint user-RB association and power allocation for uplink hybrid NOMA-OMA," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5119–5131, 2019.

[16] M. Hedayati and I. Kim, "On the Performance of NOMA in the Two-User SWIPT System," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 258–11 263, 2018.

[17] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: computation offloading as the bridge," *IEEE Netw. Mag.*, vol. 27, no. 5, pp. 28–33, 2013.

[18] A. Kiani and N. Ansari, "Edge computing aware NOMA for 5G networks," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1299–1306, 2018.

[19] Y. Wu, B. Shi, L. P. Qian, F. Hou, J. Cai, and X. Shen, "Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4811–4822, 2019.

[20] Y. Ye, L. Shi, H. Sun, R. Q. Hu, and G. Lu, "System-Centric Computation Energy Efficiency for Distributed NOMA-Based MEC Networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8938–8948, 2020.

[21] M. Eliodorou, C. Psomas, I. Krikidis, and S. Socratous, "Energy Efficiency for MEC Offloading with NOMA through Coalitional Games," in *Proc. IEEE GLOBECOM*, Waikoloa, HI, USA, 2019, pp. 1–6.

[22] S. Mao et al., "Joint Communication and Computation Resource Optimization for NOMA-Assisted Mobile Edge Computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Shanghai, China, 2019, pp. 1–6.

[23] Z. Ding, J. Xu, O. A. Dobre, and H. V. Poor, "Joint power and time allocation for NOMA–MEC offloading," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6207–6211, 2019.

[24] H. Li, F. Fang, and Z. Ding, "Joint resource allocation for hybrid NOMA-assisted MEC in 6G networks," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 241–252, 2020.

[25] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 392–407, 2019.

[26] L. P. Qian, Y. Wu, J. Ouyang, Z. Shi, B. Lin, and W. Jia, "Latency Optimization for Cellular Assisted Mobile Edge Computing via Non-Orthogonal Multiple Access," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5494–5507, 2020.

[27] M. Sheng et al., "Delay-aware computation offloading in NOMA MEC under differentiated uploading delay," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2813–2826, 2020.

[28] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "NOMA-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 244–12 258, 2018.

[29] Z. Ding, D. W. K. Ng, R. Schober, and H. V. Poor, "Delay minimization for NOMA-MEC offloading," *IEEE Signal Process. Lett.*, vol. 25, no. 12, pp. 1875–1879, 2018.

[30] J. Zhu, J. Wang, Y. Huang, F. Fang, K. Navaie, and Z. Ding, "Resource Allocation for Hybrid NOMA MEC Offloading," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4964–4977, 2020.

[31] Q.-V. Pham et al., "Coalitional games for computation offloading in NOMA-enabled multi-access edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1982–1993, 2019.

[32] Y. Ye, R. Q. Hu, G. Lu, and L. Shi, "Enhance Latency-Constrained Computation in MEC Networks Using Uplink NOMA," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2409–2425, 2020.

[33] P. Yang et al., "Latency optimization for multi-user NOMA-MEC offloading using reinforcement learning," in *Proc. IEEE 28th Wireless Opt. Commun. Conf.*, Beijing, China, 2019, pp. 1–5.

[34] L. P. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted Multi-task Multi-access Mobile Edge Computing via Deep Reinforcement Learning for Industrial Internet of Things," *IEEE Trans. Ind. Inform.*, 2020.

[35] N. Maurice, Q-V. Pham, and W-J. Hwang, "Online Computation Offloading in NOMA-based Multi-Access Edge Computing: A Deep Reinforcement Learning Approach," *IEEE Access*, vol. 8, pp. 99 098–99 109, 2020.

[36] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Commun. Mag.*, vol. 25, no. 3, pp. 116–122, 2018.

[37] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement Learning-Based Mobile Offloading for Edge Computing Against Jamming and Interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, 2020.

[38] K. N. Doan, M. Vaezi, W. Shin, H. V. Poor, H. Shin, and T. Q. Quek, "Power allocation in cache-aided NOMA systems: Optimization

and deep reinforcement learning approaches," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 630–644, 2019.
[39] M. Al-Imari, P. Xiao, M. A. Imran, and R. Tafazolli, "Uplink non-orthogonal multiple access for 5G wireless networks," in *Proc. IEEE 11th ISWCS*, Barcelona, Spain, 2014, pp. 781–785.
[40] Z. Yang, Z. Ding, P. Fan, and N. Al-Dhahir, "A general power allocation scheme to guarantee quality of service in downlink and uplink NOMA systems," *IEEE Trans. Wireless Commun.*, vol. 15, no. 11, pp. 7244–7257, Nov. 2016.
[41] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 2716–2720.
[42] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2014.
[43] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer Science & Business Media, Berlin, Germany, 2006.
[44] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, USA, 2018.
[45] H. Y. Ong, K. Chavez, and A. Hong, "Distributed deep Q-learning," *arXiv preprint arXiv:1508.04186*, 2015.
[46] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[48] A. Vaswani et al., "Attention is all you need," in *Proc. 30th Neural Inform. Process. Syst. (NIPS)*, Long Beach, CA, USA, 2017, pp. 5998–6008.
[49] C. He, Y. Hu, Y. Chen, and B. Zeng, "Joint power allocation and channel assignment for NOMA with deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2200–2210, 2019.

Van Dat Tuong received the B.S. degree in Mechatronics from Hanoi University of Science and Technology, Vietnam, in 2012, and M.S. degree in Computer Science and Engineering from Chung-Ang University, South Korea, in 2021. From 2012 to 2018, he was a Software Engineer with the Mobile R&D Center, Samsung Electronics Vietnam, Hanoi, Vietnam. From 2018 to 2021, he was a recipient of the Global Korea Scholarship sponsored by the Korean Government. He is currently pursuing his Ph.D. degree in Big Data at Chung-Ang University, South Korea. His research interests include wireless communication, mobile edge computing, reinforcement learning, and Internet of Things.

Thanh Phung Truong received the B.S. degree in Electronics-Telecommunications Engineering from Ho Chi Minh City University of Technology, Vietnam, in 2018. He is currently pursuing his Master degree in Big Data at Chung-Ang University, South Korea. From 2019 to 2020, he was a FPGA Engineer with Telecommunication Research and Development Institute, VinSmart Research and Manufacture Joint Stock Company, Vietnam. His research interests include machine learning, computing, cube-sat and wireless communication.

The-Vi Nguyen received the B.S. degree in Mathematics from University of Science, Ho Chi Minh City, Viet Nam in 2016, and M.S. degree in Computer Science and Engineering from Chung-Ang University, South Korea in 2021. He is currently pursuing Ph.D. in Big Data at Chung-Ang University, South Korea. His research interests include machine learning, optimization, and their applications in wireless communication.

Wonjong Noh received the B.S., M.S., and Ph.D. degrees from the Department of Electronics Engineering, Korea University, Seoul, South Korea, in 1998, 2000, and 2005, respectively. From 2005 to 2007, he conducted the Postdoctoral Research with Purdue University, West Lafayette, IN, USA, and the University of California at Irvine, Irvine, CA, USA. From 2008 to 2015, he was a Principal Research Engineer with the Samsung Advanced Institute of Technology, Samsung Electronics, South Korea. After that, he worked as an Assistant Professor with the Department of Electronics and Communication Engineering, Gyeonggi University of Science and Technology, South Korea, and since 2019, he has worked as an Associate Professor with the School of Software, Hallym University, South Korea. He received the Government Postdoctoral Fellowship from the Ministry of Information and Communication, South Korea, in 2005. He was also a recipient of the Samsung Best Paper God Award in 2010, the Samsung Patent Bronze Award in 2011, and the Samsung Technology Award in 2013. His current research interests include fundamental analysis and evaluations on machine learning-based 5G and 6G wireless communications and networks.

Sungrae Cho received B.S. and M.S. degrees in Electronics Engineering from Korea University, Seoul, South Korea, in 1992 and 1994, respectively, and Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002. He is a Professor with the School of Computer Science and Engineering, Chung-Ang University (CAU), Seoul, South Korea. Prior to joining CAU, he was an Assistant Professor with the Department of Computer Sciences, Georgia Southern University, Statesboro, GA, USA, from 2003 to 2006, and a Senior Member of Technical Staff with the Samsung Advanced Institute of Technology (SAIT), Kiheung, South Korea, in 2003. From 1994 to 1996, he was a Research Staff Member with Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. From 2012 to 2013, he held a Visiting Professorship with the National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. His current research interests include wireless networking, ubiquitous computing, and ICT convergence. He has served as the Organizing Committee Chair for numerous international conferences, such as IEEE SECON, ICOIN, ICTC, ICUFN, TridentCom, and the IEEE MASS, and as a Program Committee Member for conferences such as IEEE ICC, MobiApps, SENSORNETS, and WINSYS. He has been a Subject Editor for IET Electronics Letter since 2018 and an Editor for Ad Hoc Networks Journal (Elsevier) from 2012 to 2017.