IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

SPECIAL SECTION ON ADVANCES IN MACHINE LEARNING
AND COGNITIVE COMPUTING FOR INDUSTRY APPLICATIONS

Received May 3, 2020, accepted May 16, 2020, date of publication May 27, 2020, date of current version June 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997925

Online Computation Offloading in NOMA-Based Multi-Access Edge Computing: A Deep Reinforcement Learning Approach

MAURICE NDUWAYEZU[1], QUOC-VIET PHAM[2], (Member, IEEE), AND WON-JOO HWANG[3,4], (Senior Member, IEEE)

[1]Department of Information and Communication System, Inje University, Gimhae 50834, South Korea
[2]Research Institute of Computer, Information and Communication, Pusan National University, Busan 46241, South Korea
[3]School of Biomedical Convergence Engineering, Pusan National University, Yangsan 50612, South Korea
[4]Department of Information Convergence Engineering (Artificial Intelligence), Pusan National University, Busan 46241, South Korea

Corresponding author: Won-Joo Hwang (wjhwang@pusan.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MSIT) (2020-0-01450, Artificial Intelligence Convergence Research Center [Pusan National University]), and also in part Pusan National University Research Grant, 2020.

ABSTRACT One of the missions of fifth generation (5G) wireless networks is to provide massive connectivity of the fast growing number of Internet of Things (IoT) devices. To satisfy this mission, non-orthogonal multiple access (NOMA) has been recognized as a promising solution for 5G networks to significantly improve the network capacity. Considered as a booster of IoT devices, and in parallel with the development of NOMA techniques, multi-access edge computing (MEC) is also becoming one of the key emerging technologies for 5G networks. In this paper, with an objective of maximizing the computation rate of an MEC system, we investigate the computation offloading and subcarrier allocation problem in Multi-carrier (MC) NOMA based MEC systems and address it using Deep Reinforcement Learning for Online Computation Offloading (DRLOCO-MNM) algorithm. In particular, the DRLOCO-MNM helps each of the user equipments (UEs) decides between local and remote computation modes, and also assigns the appropriate subcarrier to the UEs in the case of remote computation mode. The DRLOCO-MNM algorithm is especially advantageous over the other machine learning techniques applied on NOMA because it does not require labeled data for training or a complete definition of the channel environment. The DRLOCO-MNM also does avoid the complexity found in many optimization algorithms used to solve channel allocation in existing NOMA related studies. Numerical simulations and comparison with other algorithms show that our proposed module and its algorithm considerably improve the computation rates of MEC systems.

INDEX TERMS 5G networks, deep reinforcement learning (DRL), multi access edge computing (MEC), non-orthogonal multiple access (NOMA), online computation offloading.

I. INTRODUCTION

In 5G and beyond, user equipments (UEs) are expected to run compute-intensive, latency-sensitive, and energy-hungry applications. Some examples include online gaming, virtual/augmented reality, real-time media streaming, natural language processing, online healthcare services, vehicle to vehicle applications and so on [1]–[6]. In addition, with the emerging IoT technologies and intelligent transportation systems, a huge amount of sensory data also needs high memory and strong battery-equipped devices [7]. Nevertheless, most of those UEs have some limitations like low processing capabilities and weak energy storage battery. These defects would hinder them from accomplishing the task-intensive applications that come with the mentioned applications [2], [3], [5], [7].

To cope with those challenges, MEC has been coined and developed by the European Telecommunications Standards Institute (ETSI). This MEC was developed as a new platform to provide information technology and cloud computing capabilities within the radio access network in close proximity to mobile subscribers [8], [9]. With this technology, UEs can migrate their intensive tasks to the edge of network where computation resources are sufficient to process those

The associate editor coordinating the review of this manuscript and approving it for publication was Hong-Ning Dai.

applications [10]. As opposed to mobile cloud computing, MEC is able to achieve lower latency and higher reliability and energy efficiency, which are suitable for ultra-reliable and low-latency applications in the emerging 5G networks [2], [6], [9], [11], [12].

Practically, MEC involves wireless links through which the UEs offload their tasks to the MEC server or download the results processed by it. With an exponential increase in the mobile Internet traffics over the past and the current decades [13], even some analyses show that there will be 80 billions of devices connected to the Internet by 2025, resulting in a tenfold traffic growth compared with 2016 [14], so equipping MEC with technologies which enable it to accommodate a large number of UEs is indispensably required. At this end, a number of researchers have devoted their efforts to this field to mainly solve offloading decisions and resource (communication and computation) sharing among UEs [5], [9], [15]–[19].

Pham *et al.* in [9] with an objective to minimize the system-wide computation overhead, studied a novel framework for joint computation offloading and resources allocation in MEC networks with wireless backhaul. They jointly considered offloading decision and computation resources. Yu *et al.* in [20] studied a joint subcarrier and central processing unit (CPU) time allocation for MEC. Specifically, they considered a cloudlet in an orthogonal frequency-division multiple access (OFDMA) system with multiple mobile devices where they studied a coordinate management of subcarriers and CPU of the cloudlet. Each subcarrier was allocated at most one and only one UE. Their coordinate scheduling of both subcarriers and CPU resources paid off a huge amount of energy saving. However, it could be easily inferred that the scheme resulted in a waste of bandwidth as each subcarrier is occupied by only one UE.

Many more other researchers worked on the problem of offloading decisions and resources allocation in the orthogonal multiple access (OMA) settings. Chen *et al.* in [15] deployed a deep reinforcement learning (DRL) based decentralized dynamic computation offloading strategy. They adopted deep deterministic policy gradient (DDPG) to enable each UE to leverage only local observation of the MEC system. This leverages gradually learn efficient policies for dynamic power allocation of both local execution and computation offloading in a continuous domain. Huang *et al.* in [16] with an objective to acquire an online algorithm under time-varying wireless channels, jointly optimized the task into offloading decision and wireless resource allocation to maximize the system-wide computation rate of all UEs. Li *et al.* in [21] dedicated their efforts to design a reinforcement learning (RL)-based MEC computation offloading system as a replacement of Markov decision process (MDP). This MDP can obtain an optimal policy by dynamic programming methods, which require a fixed state transition probability matrix p. However, as the number of users increases and when the environment is not explicitly defined (channel dynamism), the MDP becomes impractical. With a goal to reduce the system-wide sum cost, they first designed a Q-learning and then improved their model by using deep Q network (DQN).

In the above papers, the deployed techniques in MEC are in the OMA settings, where multiple UEs share a wireless channel by yielding one another in either time (e.g., TDMA) and frequency (e.g., OFDMA). In contrast with OMA where radio resources are allocated orthogonally to multiple users, NOMA allows multiple users to share the same resources [22]–[24]. By serving multiple users simultaneously over the same radio resources, more users can be supported, thus leading to a significant increase in the network capacity and system throughput [17], [24]. To this end, NOMA technique has been recognized as a promising solution for 5G and have attracted extensive research recently. These advantages of NOMA are nevertheless available at the expense of intra-cell interference as well as additional complexity at the receiver side. To deal with this intra-cell interference and the complexity, NOMA splits the users in the power domain based on their respective channel conditions. At the receiver side, it employs efficient multi-user detection techniques such as successive interference cancellation [23].

Kiani *et al.* in [17], as the very first attempt to reap the potential gains of NOMA in the context of MEC, proposed an edge computing aware NOMA technique. Wang *et al.* in [18] studied MEC system with multi-antenna NOMA-based computation offloading. They considered the partial offloading case, such that each user can partition the computation task into two parts for local computing and offloading. Under this setting, they minimized the weighted sum of energy consumption of all users subject to computation latency constraints. Ning *et al.* in [25] used a hybrid computation offloading framework for real-time traffic management in 5G networks. Specifically, they considered both NOMA enabled and vehicle-to-vehicle (V2V) based traffic offloading. The problem was formulated as a joint task distribution, subchannel assignment, and power allocation problem, with the objective of maximizing the sum offloading rate. It is worth noting that in our paper subchannel and subcarrier terms have the same meaning and can be used interchangeably. Gui *et al.* in [26] proposed a novel and effective deep learning (DL)-aided NOMA system, in which several NOMA users with random deployment are served by one base station. They proposed it as a remedy to the fundamental limits of the existing NOMA systems such as high computational complexity and sharply changing wireless channels. These limits make exploiting the channel characteristics and deriving the ideal allocation methods very difficult tasks. Even though this paper tried to address the issue of variability of channel environment, it still needs deep neural network (DNN) labeled training processes which also increase the complexity of NOMA systems especially when there is a large number of UEs as it is the case in 5G and IoT technologies [27], [28].

After analyzing all the improvement made on MEC, and also the benefits of NOMA as an effective solution to increase MEC capacity, we aim at investigating a NOMA-based

MEC computation offloading scheme that uses DRL and can adapt the changeability of wireless channel, and can also work in the settings without necessarily proving labeled data samples to train DNN. To the best of our knowledge, this work is the first attempt to exploiting the benefits of DRL for MC-NOMA-based MEC computation offloading where environment (channel gains) cannot be modeled. Differently from other reinforcement learning algorithms like DQN or Q-learning [29], we do not necessarily need to store Q values, and the training of DNN is accomplished following experience replay as is in [30], [31]. Recently, Yang *et al.* in [32] considered a cache-aided NOMA MEC system. They employed a long-short term memory network to predict the traffic patterns and task popularity. They also applied single- agent Q-learning algorithm for resource allocation and multi-agent Q-learning algorithm for task offloading decisions. In [33], Doan *et al.* investigated two methods for optimizing power allocation in cache-enabled NOMA systems: a divide-and-conquer-based method and a DRL-based scheme. Differently from the studies in [32], [33], our work considers multi-carrier NOMA enabled MEC systems and proposes a DRL algorithm, by which both offloading decisions and subcarrier assignment are optimized.

With an objective of maximizing the system computation rate (in terms of bits processed over a given time duration), we propose an Online Computation Offloading using DRL algorithm to solve the problem of offloading decision and subcarrier allocation in Multi-carrier NOMA-enabled MEC systems (DRLOCO-MNM). The novel contributions of our paper can be summarized as below.

1) By establishing an MC-NOMA system, we attempt to solve the problem of subchannel allocation by using DRL for the first time. We apply this technique instead of the traditional DL algorithms which require frequent parameter updating (training) such as Gui *et al.* in [26] and complete awareness of the channel environment, and other traditional optimization-based algorithms which are complex and hard to solve.
2) With the DRLOCO-MNM module, the NOMA-based MEC system can afford to accommodate considerable high capacity. In fact, NOMA is able to accommodate many UEs. Defining their offloading decisions could be obviously difficult if algorithms other than heuristics, using for instance Branch and Bound such as [34], are used. Interestingly, our proposed DRLOCO-MNM algorithm helps many UEs decide their offloading modes by considering few actions, and also assigns subcarriers without complex optimization problems. Therefore, it can easily serve more UEs by solving their computation modes (i.e., local computing at the UE or edge execution at the MEC server) and subcarrier allocation by considering only two binary actions 0 or 1 for each UE.
3) We simulate the results of our proposed DRLOCO-MNM algorithm, and the results witness that our model contributes to a remarkable increase of MEC computation rate. The system-wide weighted sum computation rate increases as the number of subchannels also increases.
4) We evaluate the performance of our DRLOCO-MNM algorithm by comparing it with OMA (TDMA)-based algorithm, optimal (exhaustive search) algorithm, edge computing and local computing algorithms. The results show that our algorithm achieves near-optimal performances and outweighs the other algorithms.
5) Moreover, with this DRLOCO-MNM module we can adjust the proportion of the UEs which can upload their computation tasks through NOMA to the MEC server and those which can run their computation locally. This is especially beneficial to NOMA as it, in some circumstances, is subject to changes of bandwidth or experiences a high level of noise.

The remaining part of this paper is organized as follows. In Section II, we describe the system model and problem formulation. In Section III, we describe in details the DRLOCO-MNM algorithm used in the MC-NOMA MEC system. Section IV is dedicated for the numerical results presentation and finally we conclude our paper in Section V.

II. SYSTEM MODEL

A. NETWORK MODEL

In our work, an MEC architecture with N UEs and one Macro-eNodeB (MeNB) is considered as shown in Fig. 1, where an MEC server is co-hosted with the MeNB. We define two sets $\mathcal{N} = \{1, \ldots, N\}$ and $\mathcal{S} = \{1, \ldots, S\}$ denoting the set of UEs and orthogonal subcarriers, respectively. Since in NOMA a subcarrier can be shared by many UEs, the received signal of the n-th UE at the MeNB contains not only desired signal but also interfering signals from co-sharing UEs.

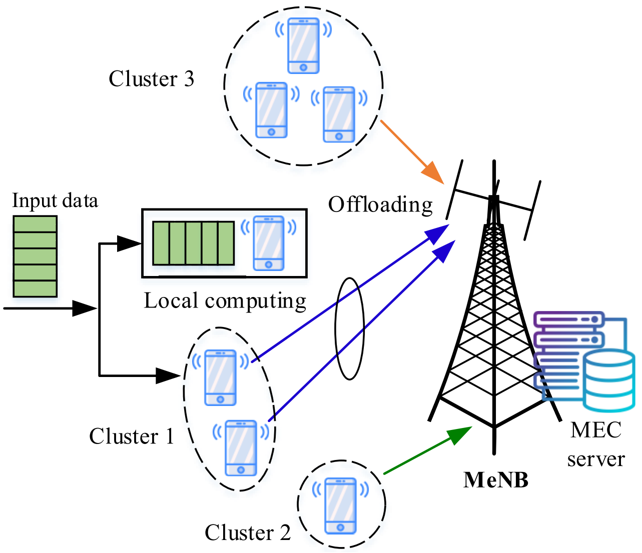


FIGURE 1. The illustration of an MC-NOMA MEC system, where more than one UE can utilize the same spectrum resource for computation offloading.

Let $\mathcal{U}_s$ denotes the set of orders of UEs sharing subcarrier s. It is assumed that each UE can utilize at most one subcarrier to offload its computations to the MEC server. We have $\mathcal{U}_s \bigcap \mathcal{U}_{s'} = \emptyset, \forall s \neq s'$, and $\bigcup_{s \in \mathcal{S}} \mathcal{U}_s = \mathcal{N}$.

B. COMMUNICATION MODEL

Let $\boldsymbol{X} = \{x_{sn} | s \in \mathcal{S}, n \in \mathcal{N}\}$ denotes the offloading decision profile. If the n-th UE does not utilize subcarrier s, $x_{sn} = 0$ whereas $x_{sn} = 1$ otherwise. Since each UE can share the uplink spectrum resource from no more than one subcarrier, we have the following constraint:

$$\sum_{s \in \mathcal{S}} x_{sn} \leq 1, \quad \forall n \in \mathcal{N}. \tag{1}$$

If $|\mathcal{U}_s| = 1, \forall s \in \mathcal{S}$, NOMA-enabled MEC becomes OMA-enabled MEC, where each subcarrier is assigned to at most one UE. From the above offloading profile, we can deduce another variable x_n for each UE to characterize whether the local computing or remote execution mode is utilized. $x_n = 1$ if the n-th UE decides to offload its computations to MEC or $x_n = 0$ when the n-th UE runs its computations locally. We can easily deduce from Eq. (1) the following equality:

$$\sum_{s \in \mathcal{S}} x_{sn} = x_n. \tag{2}$$

Let $h_{sn}(t)$ denotes the uplink channel gain between the MeNB and the n-th UE at the time frame t and p_{sn} denotes the transmit power of the n-th UE, both on subcarrier s. The channel gains on the subcarrier s are sorted in the ascending order and the bijection $b_s(\cdot)$ represents this order, where $b_s(j)$ denotes the position of UE j in the sorted sequence on subcarrier s. This is in accordance with the fact that in power-domain NOMA [23], the decoding order for uplink NOMA follows the decreasing order of the channel gains normalized by noise, while that of downlink NOMA is the increasing order [35]. In the uplink NOMA, the MeNB will start by decoding the powerful N-th UE's signals and perform SIC to cancel the resulted interference. Then it will proceed on decoding the $(N-1)$-th UE's signal and so on. Therefore, when decoding UE j's message, the signals intended for all UEs i, where $i > j$, are canceled whereas the signals of the UEs with $i < j$ are treated as noise. Without loss of generality, the received signals from UEs with $b_s(j) < b_s(i)$ is not decoded by UE i and thus is treated as noise [36]–[38]. The signal-to-interference-plus-noise ratio (SINR) of UE n on subcarrier s is expressed as follows:

$$SINR_{sn}(t) = \frac{p_{sn} h_{sn}(t)}{\sum\limits_{j \in \mathcal{J}(s,n)} p_{sj} h_{sj}(t) + n_0}, \tag{3}$$

where $\mathcal{J}(s,n)$ denotes the set of UEs, on subcarrier s, whose signals are treated as noise at the n-th UE. That noise is defined as $\mathcal{J}(s,n) = \{j \in \mathcal{U}_s : b_s(j) < b_s(n)\}$. For a more simplified presentation we encourage the interested readers to read [39] where two-user decoding scheme has been studied. This interference combines both the channel intrinsic noise power n_0, and the sum $\sum_{j \in \mathcal{J}(s,n)} p_{sj} h_{sj}(t)$ which is the interference introduced by co-subcarrier UEs. For UE n on subcarrier s, the achievable data rate is $R_{sn} = B \log_2(1 + SINR_{sn})$, where B is the bandwidth of an orthogonal subcarrier.

C. COMPUTATIONAL MODEL

Each UE n has a computational task of $I_n = (D_n, C_n)$ where D_n denotes the input data size (in bits) and C_n denotes the computation workload (CPU cycles per bit) of UE n.

In the local computation mode, i.e., $x_n = 0$, the UE relies on its capability and processes its applications. Let f_n^l denotes the computing speed (CPU cycles per second) of UE n where the superscript l symbolizes the local computation mode. The task I_n completion time can be computed as $T_n^l = (D_n \times C_n)/f_n^l$ and it can be easily shown that the computation rate achieved by the n-th UE in the local mode depends on its CPU computing speed and is expressed as follows:

$$r_n^l = \frac{f_n^l}{C_n}. \tag{4}$$

In the case of remote computation, i.e., $x_n = 1$, the processor's computing speed of the n-th UE is supposed to be unable to accomplish the task given, therefore it seeks help on the MEC server to run its heavy computations. Consequently, it uses one of the subcarriers to send its task to MEC server. The computation rate of the n-th UE in offloading mode can be expressed as follows:

$$r_n^o(t) = B \sum_{s \in \mathcal{S}} x_{sn} \log_2 \left(1 + \frac{p_{sn} h_{sn}(t)}{\sum\limits_{j \in \mathcal{J}(s,n)} p_{sj} h_{sj}(t) + n_0} \right). \tag{5}$$

The metric "computation rate" has been considered and evaluated in many existing studies on MEC, e.g., computation rate maximization in wireless powered MEC system with partial offloading and binary offloading [40], [41], and in UAV-enabled wireless powered MEC system with both partial and binary offloading [42]. It is worth mentioning that other performance metrics can be used in our DRL framework, such as, energy efficiency and completion latency [43], and computation overhead [4], [9].

D. PROBLEM FORMULATION

We aim to maximize the system-wide computation rate of the MC-NOMA based MEC network in a tagged time frame t. We define the matrix of channel gains at the time frame t as $\boldsymbol{H}(t) = [\boldsymbol{h}_1(t), \boldsymbol{h}_2(t), \ldots, \boldsymbol{h}_S(t)]$ and each element of $\boldsymbol{H}(t)$ is defined as $\boldsymbol{h}_s(t) = [h_{s1}(t), h_{s2}(t), \ldots, h_{sN}(t)]$ that denotes the time-varying wireless channel gain of the N UEs on subcarrier s at the time frame t. Thus, the weighted sum computation rate of the MEC system is denoted as

$$Q(\boldsymbol{H}, \boldsymbol{X}) \triangleq \sum_{n=1}^{N} w_n \left((1 - x_n) r_n^l + x_n r_n^o \right), \tag{6}$$

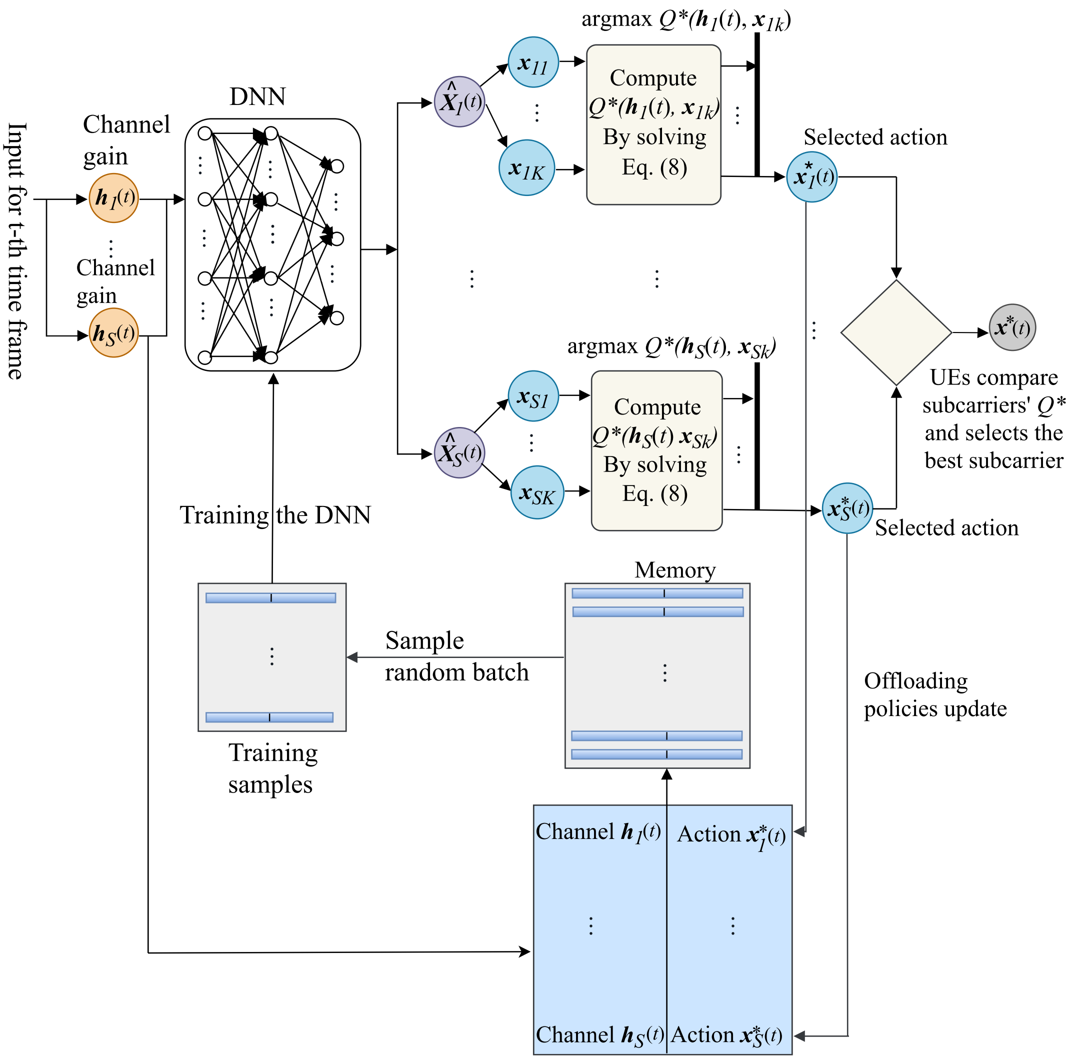


FIGURE 2. The schematics of the proposed DRLOCO-MNM algorithm.

where $w_n > 0$ denotes the weight assigned to the n-th UE and r_n^o is the computation rate of the n-th UE in the chosen subcarrier s out of all other subcarriers calculated using Eq. (5). For each channel realization $\boldsymbol{H}$, we are interested in maximizing the total weighted sum computation rate as follows:

$$Q^*(\boldsymbol{H}, \boldsymbol{X}) = \max_{X} Q(\boldsymbol{H}, \boldsymbol{X})$$
$$\text{subject to} \sum_{s \in \mathcal{S}} x_{sn} \leq 1, \quad \forall n \in \mathcal{N},\ s \in \mathcal{S}. \quad (7)$$

The primordial step to solve this system-wide computation rate is to determine the value of x_{sn} and deduce x_n, which are subcarrier allocation and offloading decision, respectively. To solve it, we adopt the algorithm in [16] and modify it so that our proposed algorithm not only solves the offloading decision as it did in the original work but also assigns subcarriers to UEs, along with other tasks in the MC-NOMA based MEC settings considered in our work.

III. THE DRLOCO-MNM ALGORITHM

Deep reinforcement learning shares the same basic concepts with reinforcement learning in that it is also an agent-environment interaction process. However, reinforcement learning becomes less effective when dealing with actual complicated problems of high dimensional-state and action spaces. To overcome this challenge, DRL is built in two components: offline DNN and online Q-leaning [44]. In the offline phase, a DNN is constructed, which can infer for each state-action pair its Q value to be used for the online phase. Sufficient training data is needed for the offline DNN construction. In the online phase, deep Q-learning is adopted for the action selection (i.e., the ϵ-greedy policy, in our case this is the computation of $Q^*(\boldsymbol{H}, \boldsymbol{X})$) and Q value update. In an online learning case, the agent gradually gathers experience in the environment. The online offloading property of the DRL lies on the fact that its interaction with the channel environment is on the go basis, without prior training of data [44], [45].

The detailed schematic diagram of our DRLOCO-MNM algorithm can be seen in Fig. 2. It is composed of three main consecutive stages namely offloading action generation, offloading policies update, and subcarrier selection for UEs. At each time frame t, the DNN receives S subcarriers' channel gain vectors in the form of $\boldsymbol{H}(t)$ as inputs and each of its elements generates relaxed offloading actions $\widehat{\boldsymbol{X}}_s(t)$. As the output of the DNN, the offloading action value of each UE is not a binary 0 or 1. This relaxed value is represented by a parameterized function $f_{\theta(t)}$ such that $\widehat{\boldsymbol{X}}_s(t) = f_{\theta(t)}\,(\boldsymbol{h}_s(t))$, where

$$\widehat{\boldsymbol{X}}_s(t) = \{\hat{x}_{sn}(t)|\hat{x}_{sn}(t) \in [0, 1], n = 1, \ldots, N\},$$

and $\hat{x}_{sn}(t)$ denotes the n-th entry of $\widehat{\boldsymbol{X}}_s(t)$. By parameterized we mean that its values depend on the parameters $\theta(t)$ of the DNN and those parameters change when experiment replay training is performed which also means that the relaxed values change whenever we train the DNN.

This relaxed value is then quantized following a quantization function g_k defined as

$$g_k : \widehat{\boldsymbol{X}}_s \mapsto \{\boldsymbol{x}_{sk}|\boldsymbol{x}_{sk} \in [0, 1]^N, k = 1, \ldots, K\}$$

to generate multiple binary actions among which we choose the best offloading actions $\boldsymbol{x}_s^* \in [0, 1]^N$. The quantization adopted is the order preserving quantization (OP) method that produces at most N binary offloading actions and has the advantage of balancing the algorithm complexity and performance (the more number of offloading actions K, the more the accuracy of learning but the more the complexity). Also more details on this quantization process and OP quantization method can be found in [16].

Each of the subcarriers goes through the same process: Apply the channel gain to DNN, compute the optimal reward $Q^*(\boldsymbol{h}_s, \boldsymbol{x}_{sk})$ and then consider the corresponding action as the optimal one $\boldsymbol{x}_s^*$. Then we deduce the reinforcement learning policy as

$$\pi : \boldsymbol{h}_s \mapsto \boldsymbol{x}_s^*. \tag{8}$$

This optimal reward is the highest computation rate of the action k, out of all the other K actions, computed over all the UEs.

All the channel gains $\boldsymbol{h}_s(t)$ of $\boldsymbol{H}(t)$ at the time frame t along with the best action $\boldsymbol{x}_s^*$ are stored in the memory and will be used to train the DNN after some interval of time δ. The n-th UE compares the possible reward in terms of computation rate it may benefit from using a subcarrier s and chooses accordingly. If the n-th UE cannot benefit from all the subcarriers (the optimal actions of the n-th UE at time frame t seen in all subcarriers is a binary 0), the offloading mode will be local computation. By contrast, if it realizes an optimal reward in a subcarrier s (at least one optimal action of the n-th UE at time frame t seen in all subcarriers is a binary 1), comparatively to the other subcarriers, it chooses that subcarrier and offloads its computations to the MEC server. As it can be seen in Fig. 3, at the time frame t, each subcarrier channel gain shall generate an offloading decision and rewards for each UE. Then each UE shall choose one of the subcarriers by comparing all the possible rewards in those subcarriers. If in all subcarriers the offloading decisions are all zero for a particular n-th UE, that user shall solve its applications locally. By contrast, if at least a binary 1 in the offloading decisions is found in one subcarrier or more according to Fig. 3, the user shall use the subcarrier with the highest reward and send its computations to the MEC server. The detailed process can also be read in Alg. 1.

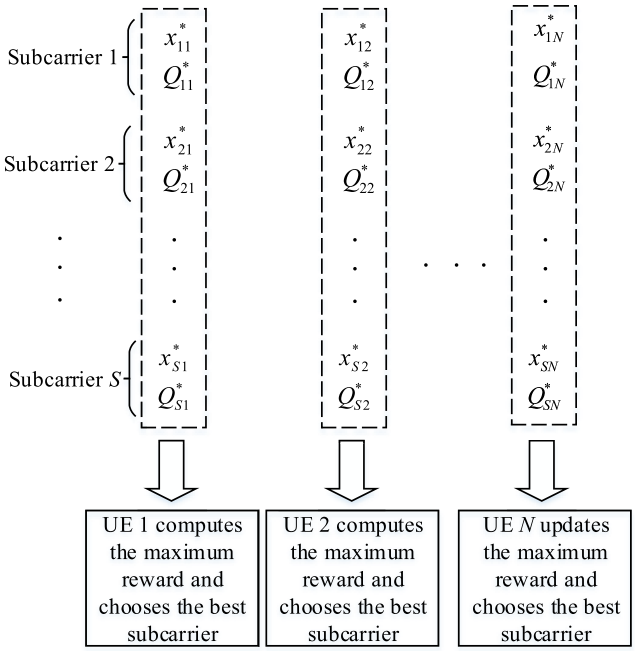


FIGURE 3. Subcarrier selection process.

Based on the steps of our DRLOCO-MNM algorithm, the process involved and the output desired, we devised the Alg. 1. The algorithm runs in one time frame and runs for all subchannels. Its inputs are the channel gains of UEs on multiple subcarriers and the output is the offloading decisions of UEs and their corresponding allocated subcarriers in the case of remote computation. The proposed algorithm runs as follows:

- Firstly, the algorithm initializes the DNN with random parameters θ_1, where the subscript 1 means initial time frame, and empties the DNN memory R. It also sets the iteration number, which is the number of time frames it runs, and specifies the number of subcarriers to be used and the time interval after which the DNN memory shall be trained.
- Secondly, the algorithm goes through the actual process of generating outputs by running from 1 to a maximum M times (line 4 up to 26). In a typical time frame, the algorithm connects the DNN to a number of channel gain vectors of different subcarriers (line 5 to 14), and each produces $\widehat{\boldsymbol{X}}_s$. Then, the relaxed values are quantized into K offloading actions, each k having N binary numbers. Each user is assigned either 0 or 1, which defines

Algorithm 1 The DRLOCO-MNM Algorithm to the Offloading Decision and Subcarrier Allocation Optimization in MC-NOMA MEC Systems

```
1:  Input: Channel gains $\boldsymbol{H}(t)$ at the time frame $t$.
2:  Output: Offloading action $x_{sn}, \forall s \in \mathcal{S}, n \in \mathcal{N}$, and corresponding allocated subcarriers, both at the time frame $t$.
3:  Initialize the DNN with random parameters $\theta_1$ and empty the memory $R$. Set the iteration number $M$, number of subcarriers $S$, and the training interval $\delta$.
4:  for $t = 1, \ldots, M$ do
5:      for $s = 1, \ldots, S$ do
6:          Generate relaxed actions $\widehat{\boldsymbol{X}}_s(t) = f_{\theta(t)}\left(\boldsymbol{h}_s(t)\right)$.
7:          Quantize $\widehat{\boldsymbol{X}}_s(t)$ into $K$ binary actions $g_k\left(\widehat{\boldsymbol{X}}_s(t)\right)$.
8:          Compute $Q^*\left(\boldsymbol{h}_s(t), \boldsymbol{x}_{sk}\right)$ for all $\{\boldsymbol{x}_{sk}, k = 1, \ldots, K\}$.
9:          Select $\boldsymbol{x}_s^*(t) = \underset{k=1,\ldots,K}{\operatorname{argmax}} Q^*\left(\boldsymbol{h}_s(t), \boldsymbol{x}_{sk}\right)$.
10:         Update the memory by adding $\left(\boldsymbol{h}_s(t), \boldsymbol{x}_{sk}^*\right)$ in $R$.
11:         for $n = 1, 2, \ldots, N$ do
12:             Compute the $Q^*\left(h_{sn}(t), x_{sn}^*(t)\right)$.
13:         end for
14:     end for
15:     for $n = 1, \ldots, N$ do
16:         if $x_{sn}^*(t) = 0$ for all $s \in \mathcal{S}$ then
17:             $x_n = \sum_{s \in \mathcal{S}} x_{sn} = 0$: local computation mode.
18:         else if $x_{sn}^*(t) \neq 0$ for at least one subcarrier then
19:             $x_n = 1$ and $s^* = \underset{s \in \mathcal{S}}{\operatorname{argmax}} Q^*\left(h_{sn}(t), x_{sn}^*\right)$.
20:         end if
21:     end for
22:     if $t$ mod $\delta = 0$ then
23:         Uniformly sample a batch of data set $\{(\boldsymbol{h}_{s\tau}, \boldsymbol{x}_{s\tau}^*) | \tau \in \tau(t)\}$ from the memory.
24:         Train the DNN with $\{(\boldsymbol{h}_{s\tau}, \boldsymbol{x}_{s\tau}^*) | \tau \in \tau(t)\}$ and update $\theta(t)$ using the Adam algorithm.
25:     end if
26: end for
```

its computation mode. Among those different quantized values, the algorithm selects the best ($\boldsymbol{x}_s^*$) based on the calculated reward. It computes the weighted sum computation rate that would be incurred by adopting each $\boldsymbol{x}_k$ and then compares them. It then selects the best action as the one corresponding to the highest return.

- Thirdly, the algorithm updates the memory with that optimal action and the current channel gain $\boldsymbol{h}_s$.
- Fourthly, the algorithm computes and stores the returns of each UE on its action in the optimal quantized value $\boldsymbol{x}_s^*$ (lines 11 to 13) and then the loop which runs through all subcarriers terminates.
- Fifthly, the algorithm goes through the returns of each UE computed in all subcarriers and compares them to choose the best return value and its corresponding subcarrier (lines 16 to 22) according to the concept shown in Fig. 3. At this stage, the subcarrier is allocated. As the optimal actions have been also stored along with the optimal returns of all UEs, the algorithm checks the action values binary) for each user and if they are all zeroes, the n-th UE handles its computations locally, but if at least one binary 1 is present, the UE uses the remote computation mode.
- The last stage of the algorithm is the training of the DNN memory (lines 22 till 25). It does so by retrieving a batch of channel gains and corresponding optimal values, which have been stored in the memory at each channel gain realization. The output of the training is the parameter of DNN θ that is used to train the next time frame.

With this training, the DNN runs from the past channel to gain the optimal action values and then becomes smarter. Therefore in the next realization of the channel gain, the DNN shall generate better more actions to improve the return (computation rate).

IV. PERFORMANCE SIMULATIONS

A. DNN MEMORY SETTINGS AND TRAINING

The DNN of the DRL we used is composed of one input layer, two hidden layers, and one output layer. The first and second hidden layers have 120 and 80 hidden neurons, respectively. We implemented the DRLOCO-MNM algorithm in Python version 3.6.4 with TensorFlow 1.4. The batch size defined as Γ, was set to be 128, 256, and as high as 512 according to the number of subcarriers used so that it can train as many samples as they are stored in memory. The memory size was set to be 1024. Our algorithm updates the memory S times in each time frame. In order to avoid the risk of training the memory (changing the parameters $\theta(t)$) in the middle of time frame t, a risk that could cause some subcarriers to be treated at different θ parameters, we set the training interval to be proportional to the number of subcarriers employed as $\delta = V \times S$, where V is a constant.

The time-varying channel gain we used $h_s(t)$ is generated according to $h_{sn}(t) = \bar{h}_{sn}\alpha_{sn}(t)$. Where $\alpha_{sn}(t)$ is an independent random channel fading factor distributed exponentially around a unit mean. In the same expression, $\bar{h}_{sn}$ is the average channel gain that follows the free space path loss model $\bar{h}_{sn} = A_d \left(\frac{3.10^8}{4\pi f_c d_n}\right)^{de}$, where the A_d, f_c, de, and d_n are the antenna gain, carrier frequency, path loss exponent and distance from the n-th UE to the MeNB, respectively. In the experiment we used $A_d = 4.11, f_c = 915$ Mhz and $d_e = 2.8$. The value of d_n used is in the range of (2.5, 5.2) meters with uniform distribution [16].

B. COMPUTATION RATE PERFORMANCE

Now each UE has decided its computation mode and the offloading UEs have been allocated subcarriers to migrate their heavy computations to the MEC server. The task to be accomplished by each UE is done by either the UE locally or by the MEC server remotely.

The computation task assigned to the UEs is the face recognition application, where the task I is set as $I = (420, 1000)$; the computation input data size (420 in kB) and the total required number of CPU cycles in Megacycles, respectively [9], [46]. The CPU computational capacity f_n^l of UE is set to be 1.0 GHz. The noise power $n_0 = -100$ dBm and the transmit power is set to be 20 dBm for all UEs [9]. The system bandwidth is 10 MHz and subcarrier bandwidth is assigned according to the number of subcarriers we use. For example, if the number of subcarriers is 5 each of those subcarriers shall have 2 MHz. With this system bandwidth, we are expecting more than 10 UEs to benefit from it and send their application to MEC server because they can share subcarriers. However, if OMA is used, at maximum only 10 UEs can be deployed in this system. At this end, the capability of MC-NOMA to increase system capacity is reiterated.

Our goal is to maximize the system-wide computation rate, which is the sum of computation rates of all UEs. We start by plotting in Fig. 4a the computation rates of different subcarriers when the time frame is 100 with a rolling interval of 50. We plot it on a short time frame scale to show that the computation rates reach stability after some time frames. The 'xCh' on the legends, stands for the number of channels or subchannels used by the MC-NOMA system. In Fig. 4b, we then plot the computation rates under different subcarriers when the number of time frames is 3000 and rolling interval is 100. From that figure, we can see clearly that as we increase the number of subcarriers, the computation rates increase. We also notice that the computation rates have already stabilized to a certain value at time frame of 3000. Moreover, as the sum computation rates are the results of comparison among different channel gains, we have some fluctuations in the convergence of the optimal computation rate especially when the number of subchannels of the NOMA-based MEC system increases. However, what is interesting, and as the main goal of our MC-NOMA based MEC is to have improved computation rates by increasing the number of subchannels.

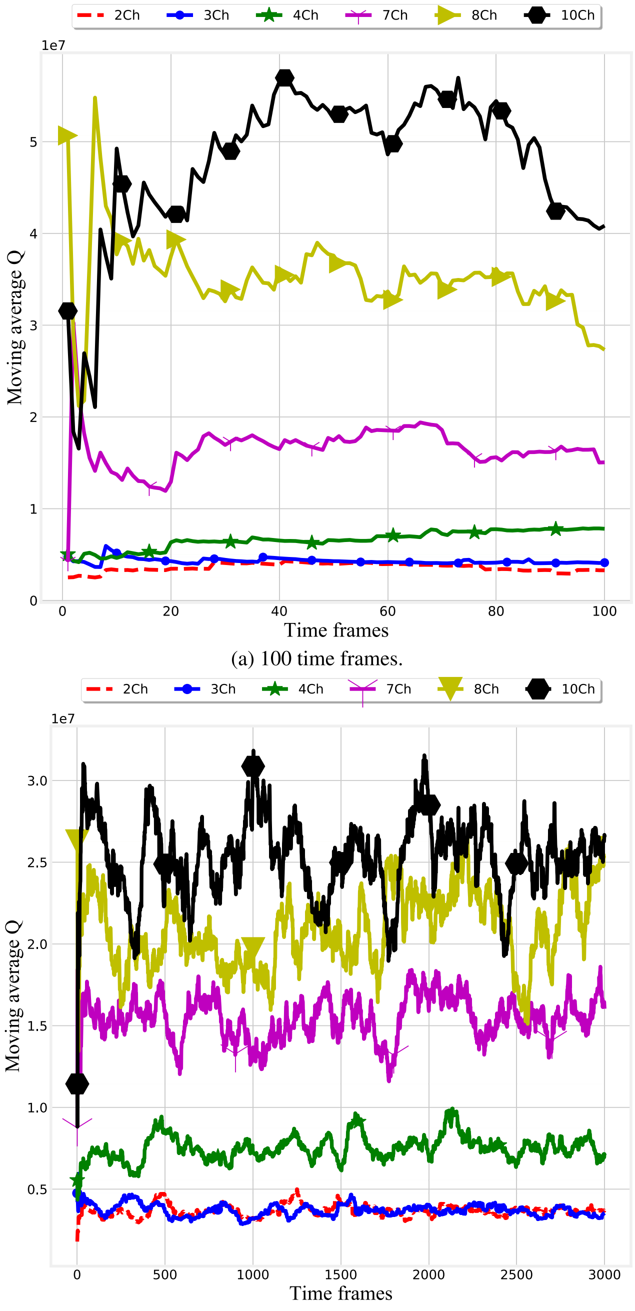
C. COST FUNCTIONS

The loss function used to train the DNN (updating the $\theta(t)$ parameters) is defined as an average cross-entropy loss as:

$$L(\theta(t)) = -\frac{1}{|\tau(t)|}\sum_{\tau\in\tau(t)}\Big(\left(\boldsymbol{x}_\tau^*\right)^\top \log f_{\theta(t)}(\boldsymbol{h}_{s\tau}) + \left(1-\boldsymbol{x}_\tau^*\right)^\top \log\left(1-f_{\theta(t)}(\boldsymbol{h}_{s\tau})\right)\Big),$$

where $|\tau(t)|$ denotes the size of $\tau(t)$, the superscript $\top$ indicates the transpose operator, and the log function denotes the element-wise logarithmic operation of a vector. This loss function applies the ADAM algorithm and we direct readers to read [47] for a more detailed explanation.

Our proposed algorithm has the advantage of increasing the convergence speed when training DNN and this speed increases as the number of subcarriers of our MC-NOMA system also increases. This can be clearly seen from the cost functions plotted in Fig. 5. The training steps required to converge reduce as we increase the number of subchannels used. We can visualize that the convergence is reached at 150 training steps in the case of 1 subchannel, about only 50 training steps for 2 subchannels, and about only 10 training steps for the 10 subchannels. These observations justify the reason why our DRLOCO-MNM algorithm learns fast. Intuitively, this fast convergence is attributed to the fact that the DNN memory R of the DRLOCO-MNM records many



(a) 100 time frames.

(b) 3000 time frames.

FIGURE 4. Moving average of the system-wide weighted sum of computation rates Q of different subcarriers.

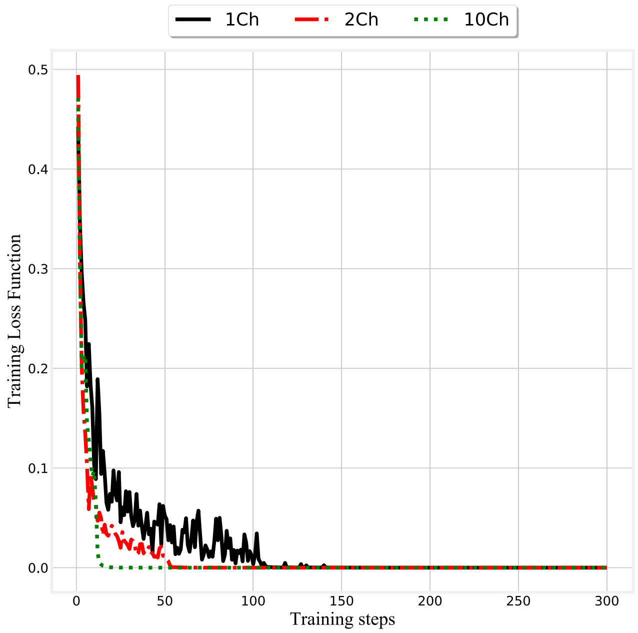


FIGURE 5. Training loss function of DNN for different subcarriers when time frames is 300.

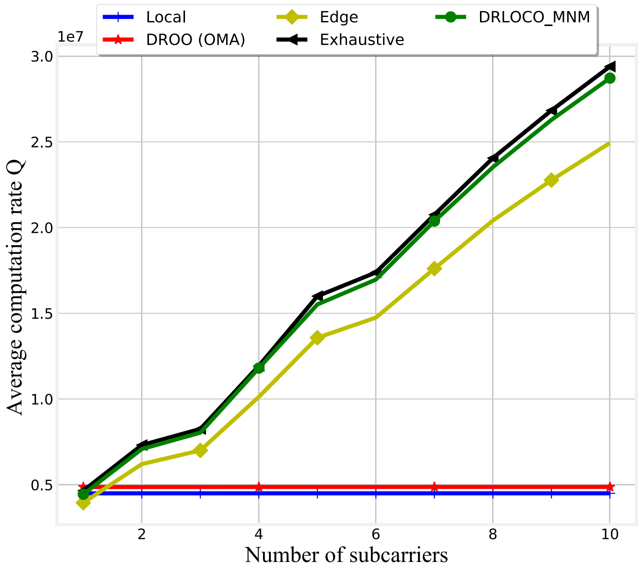


FIGURE 6. Comparisons of computation rate performances of different Algorithms.

different optimal offloading actions $\boldsymbol{x}_s^*$ of different channel gains in a single time frame; therefore, the proposed algorithm realizes the optimal offloading policy quickly. And these optimal offloading policies are applied to every $\boldsymbol{h}_s(t)$ element of $\boldsymbol{H}(t)$. On the Fig. 5, the learning steps are $1/10$ of the total time frames used. This is a result of the training interval (δ) used which is 10, 20 and 100 for single channel, 2 channels, and 10 channels, respectively.

D. COMPUTATION RATE PERFORMANCE COMPARISONS OF DIFFERENT ALGORITHMS

In Fig. 6, we compare our DRLOCO-MNM algorithm with four different other algorithms. We first compare with another DRL based algorithm, namely DROO, but with OMA (TDMA) settings [16]. Then, inorder to assess the gain of using DRL-based algorithm, we also compare our algorithm against three non-DRL based representative benchmark algorithms:

- Exhaustive search: In this algorithm we exhaustively enumerate all the possible combinations of UEs' computation modes and subcarrier allocation and then output the best performer in terms of computation rate. Here, if $S = 1$ we would have a binary choice; all the users compute either locally or remotely by using a single subcarrier. If $S = 2$, all UEs would have three options namely local computation, remote computation with subcarrier 1 or remote computation with subcarrier 2. Thus we choose the optimal option among all those possible for all the UEs.
- Edge computing: In this algorithm we presume that all UEs are incapable of carrying out their computations therefore we force them to offload their computations to the edge (MEC), i.e., $x_n = 1, n = 1, \ldots, N$.
- Local computing: In this algorithm we presume that all UEs have a sufficient computation capability therefore we force them to carry out their computations locally by themselves, i.e., $x_n = 0, n = 1, \ldots, N$.

We simulate the performances of those different algorithms under varying number of subcarriers. Because a high number of users would render our simulation prohibitive, we limit our simulation to 3 UEs. Each point in the figure is an average computation rate calculated over 1000 different wireless channel realizations (1000 time frames). With our DRLOCO-MNM, offloading UEs communicate their computations simultaneously by using different subcarriers differently from the DROO (OMA) which sequences its users over time on a single carrier. Thus from the Fig. 6 the computation rate of our algorithm increases with the increase in the number of subcarriers while that of the DROO remains constant.

The same figure shows that our algorithm can achieve a near-optimal performance. The computation rate of our DRLOCO-MNM algorithm and that of the optimal algorithm (exhaustive search) are quite similar. When compared to the edge, our algorithm tremendously outperforms it. The first reason for this is because all UEs in the edge computing migrate their computations to MEC regardless of the channel conditions, i.e., whether the channel gain is low or high, therefore they miss the leverage of their computation capability when the channel environment is unfavorable. The second reason is a high co-channel interference. This results from the fact that all UEs need a subchannel to send their data to MEC.

The weighted sum computation rate achieved by the local computing algorithm is the lowest of the four. Indeed it is because the performance of this algorithm depends on the UE's computation capability f_n^l which is lower compared to that of MEC. However, it is also worth noting that the UEs forced to run their computations locally miss to enjoy the high computations that the edge offers when the channel environment would be favorable for them. We also remind that the computation rate is independent of the subcarriers used as there is no channel needed. In fact, this comparison of our algorithm against the DROO and the other non-DRL algorithms shows that it can significantly increase computation rate when the number of subcarriers is increased and that it can adapt to the dynamism of wireless channel environment.

E. PROPORTION OF UEs IN LOCAL COMPUTATION

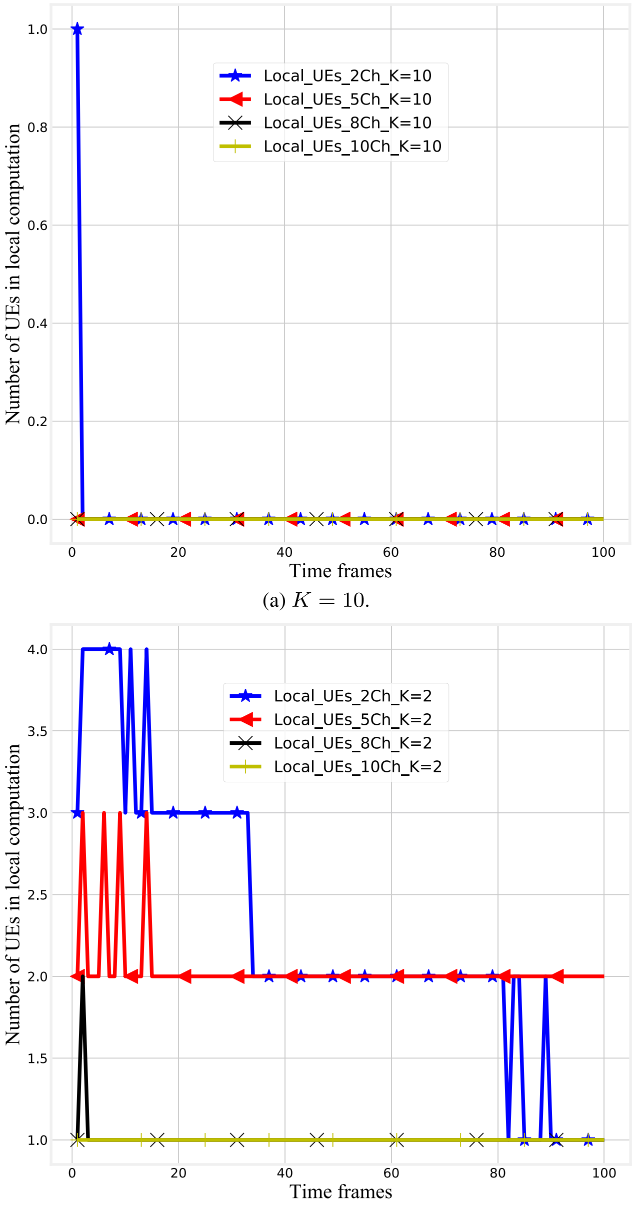
The proportion of UEs computing their applications locally is relatively lower than those sending their computations to the MEC server. That proportion depends highly on the number of subcarriers used and the initial level of quantization (K) of the relaxed output $(\widehat{X})$ from DNN. As it can be seen in Fig. 7a, where K is 10, the number of UEs in local mode is zero except when we use 2 subcarriers where we have only one instance of UE in local mode. By contrast, when we decrease the value of K down to 2 as in Fig. 7b, the number of UEs computing locally increases noticeably reaching even four in the case of 2 subcarriers but only 1 for 10 subcarriers. The intuition behind the decrease in the local UEs as we increase the number of subcarriers is because UEs have more option of subcarriers to choose from therefore a few of them compute locally. Moreover, the UEs in local mode increase following a decrease in K level because the DRLOCO-MNM algorithm does not explore many combinations (actions) of UEs, thus resulting in many zeros (the UEs in local mode have 0 in all subcarriers). This is another advantage of our DRLOCO-MNM algorithm because it gives option to regulate the number of UEs on the available number of subcarriers (accordingly bandwidth size).

F. COMPUTATIONAL COMPLEXITY

The complexity of our algorithm in terms of time of execution depends highly on the offloading decision and subchannel allocation stages. Table 1 compares the CPU times of the different algorithms. We can easily notice that the time complexity of exhaustively enumerating all the possible offloading modes is the highest of the three algorithms. The high computation rate that can be achieved by that algorithm (optimal) comes with the cost of time. Obviously this algorithm would be impractical with a high number of users. In addition, this optimal algorithm is also unfavorable with MC-NOMA because its CPU time increases when the number of subcarriers also increases as it can be seen in the table. By contrast, our algorithm which uses MC-NOMA has another advantage of saving CPU time when it uses more subcarriers. This is since the training interval as has been defined in Subsection A. The more the number of subchannels, the fewer frequencies of training. When the CPU time

TABLE 1. Comparison of time complexity of different algorithms. The times are measured in seconds.

Algorithm	CPU time		DNN train time
DRLOCO-MNM	1SC	5.9×10^{-2}	1.8×10^{-3}
	2SC	5.2×10^{-2}	
	3SC	5.1×10^{-2}	
	5SC	4.9×10^{-2}	
	10SC	4.4×10^{-2}	
Exhaustive search	1SC	8.8×10^{-2}	no DNN
	2SC	1.3×10^{-1}	
	3SC	2.1×10^{-1}	
	5SC	3.2×10^{-1}	
	10SC	5.1×10^{-1}	
DROO (OMA)	3.4×10^{-2}		2×10^{-3}



(a) $K = 10$.

(b) $K = 2$.

FIGURE 7. Proportion of UEs in local computation for different numbers of subcarriers.

of our algorithm is compared to that of the DROO that uses OMA (TDMA) we observe that our algorithm tends to be slightly more time complex than DROO. This time complexity discrepancy between the SC-NOMA and DROO (OMA) algorithms, 5.9×10^{-2} against 3.4×10^{-2}, is the result of the fact that our algorithm requires time to decide a subchannel in addition to the CPU time required to decide offloading status. However, the time required to train DNN is almost the same in both algorithms, 1.8×10^{-3} seconds to train the DNN in our algorithm is approximately the same as 2×10^{-3} seconds of the DROO algorithm. We opt not to include edge and local computation algorithm in the complexity analysis because they do not run the process of choosing the computation mode, a process which is key to the time complexity.

V. CONCLUSION

In this paper, we considered an online algorithm to maximize the weighted sum computation rate of an MC-NOMA enabled MEC network with binary computation offloading, and proposed a DRL-based algorithm, DRLOCO-MNM, to solve both the computation mode and subcarrier allocation. Thus instead of requiring manually labeled data training, the proposed algorithm can learn by experience replay to improve the computation offloading policy. Simulation results showed that our algorithm can achieve near-optimal results and significantly achieves higher computation rates as compared to the OMA (TDMA) based algorithm scheme. Also, the convergence speed of our DRLOCO-MNM algorithm improved. This improvement is a result of the fact that the proposed algorithm realizes many optimal offloading actions in one time frame, in addition to order-preserving quantization and adaptability of the algorithm in setting the DNN parameters by experience replay.

REFERENCES

[1] X. Qiao, P. Ren, S. Dustdar, L. Liu, H. Ma, and J. Chen, "Web AR: A promising future for mobile augmented reality—State of the art, challenges, and insights," *Proc. IEEE*, vol. 107, no. 4, pp. 651–666, Apr. 2019.

[2] Q.-V. Pham, F. Fang, V. Nguyen Ha, M. Jalil Piran, M. Le, L. Bao Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and State-of-the-Art," 2019, *arXiv:1906.08452*. [Online]. Available: http://arxiv.org/abs/1906.08452

[3] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proc. IEEE*, vol. 107, no. 8, pp. 1717–1737, Aug. 2019.

[4] Q.-V. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access*, vol. 6, pp. 75868–75885, 2018.

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

[6] Q.-V. Pham, S. Mirjalili, N. Kumar, M. Alazab, and W.-J. Hwang, "Whale optimization algorithm with applications to resource allocation in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4285–4297, Apr. 2020.

[7] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[8] M. Patel *et al.*, "Mobile-edge computing introductory technical white paper," Mobile-Edge Comput. Ind. Initiative, Sophia Antipolis, France, White Paper, Sep. 2014. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

[9] Q. Pham, L. B. Le, S. Chung, and W. Hwang, "Mobile Edge Computing With Wireless Backhaul: Joint Task Offloading and Resource Allocation," *IEEE Access*, vol. 7, pp. 16444–16459, 2019.

[10] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

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

[12] M. Zeng and V. Fodor, "Energy minimization for delay constrained mobile edge computing with orthogonal and non-orthogonal multiple access," *Ad Hoc Netw.*, vol. 98, Mar. 2020, Art. no. 102060.

[13] J. G. Andrews, "What will 5G be?" *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.

[14] G. Zhu, Y. Wang, and K. Huang, "Broadband analog aggregation for low-latency federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 491–506, Jan. 2020.

[15] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," 2018, *arXiv:1812.07394*. [Online]. Available: http://arxiv.org/abs/1812.07394

[16] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, early access, Jul. 24, 2019, doi: 10.1109/TMC.2019.2928811.

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

[18] F. Wang, J. Xu, and Z. Ding, "Multi-antenna NOMA for computation offloading in multiuser mobile edge computing systems," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2450–2463, Mar. 2019.

[19] 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, Mar. 2018.

[20] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and CPU time allocation for mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.

[21] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, Apr. 2018, pp. 1–6.

[22] Q.-V. Pham and W.-J. Hwang, "α-Fair resource allocation in non-orthogonal multiple access systems," *IET Commun.*, vol. 12, no. 2, pp. 179–183, Jan. 2018.

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

[24] Q.-V. Pham, H. T. Nguyen, Z. Han, and W.-J. Hwang, "Coalitional games for computation offloading in NOMA-enabled multi-access edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1982–1993, Feb. 2020.

[25] Z. Ning, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Joint computation offloading, power allocation, and channel assignment for 5G-enabled traffic management systems," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3058–3067, May 2019.

[26] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8440–8450, Sep. 2018.

[27] S. Wang, T. Lv, and X. Zhang, "Multi-agent reinforcement learning-based user pairing in multi-carrier NOMA systems," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Shanghai, China, May 2019, pp. 1–6.

[28] 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, Oct. 2019.

[29] P. Yang, L. Li, W. Liang, H. Zhang, and Z. Ding, "Latency optimization for multi-user NOMA-MEC offloading using reinforcement learning," in *Proc. 28th Wireless Opt. Commun. Conf. (WOCC)*, Beijing, China, May 2019, pp. 1–5.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning.," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[31] B. Luo, Y. Yang, and D. Liu, "Adaptive Q-Learning for data-based optimal output regulation with experience replay," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3337–3348, Dec. 2018.

[32] Z. Yang, Y. Liu, Y. Chen, and N. Al-Dhahir, "Cache-aided NOMA mobile edge computing: A reinforcement learning approach," 2019, *arXiv:1906.08812*. [Online]. Available: http://arxiv.org/abs/1906.08812

[33] K. N. Doan, M. Vaezi, W. Shin, H. V. Poor, H. Shin, and T. Q. S. Quek, "Power allocation in cache-aided NOMA systems: Optimization and deep reinforcement learning approaches," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 630–644, Jan. 2020.

[34] X. Yang, X. Yu, H. Huang, and H. Zhu, "Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems," *IEEE Access*, vol. 7, pp. 117054–117062, 2019.

[35] Z. Ding, X. Lei, G. K. Karagiannidis, R. Schober, J. Yuan, and V. K. Bhargava, "A survey on non-orthogonal multiple access for 5G networks: Research challenges and future trends," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 10, pp. 2181–2195, Oct. 2017.

[36] X. Zhang, Y. Chen, J. Zhang, Y. Lei, C. Shen, and G. Zhu, "Characterization of SINR region for multi-cell downlink NOMA systems," in *Proc.. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[37] Y. Gao, B. Xia, K. Xiao, Z. Chen, X. Li, and S. Zhang, "Theoretical analysis of the dynamic decode ordering SIC receiver for uplink NOMA systems," *IEEE Commun. Lett.*, vol. 21, no. 10, pp. 2246–2249, Dec. 2017.

[38] 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.

[39] W. Wu, F. Zhou, R. Q. Hu, and B. Wang, "Energy-efficient resource allocation for secure NOMA-enabled mobile edge computing networks," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 493–505, Jan. 2020.

[40] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[41] M. Zeng, R. Du, V. Fodor, and C. Fischione, "Computation Rate Maximization for Wireless Powered Mobile Edge Computing with NOMA," in *Proc. IEEE 20th Int. Symp.*, Washington, DC, USA, Jun. 2019, pp. 1–9.

[42] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[43] Z. Yang, C. Pan, J. Hou, and M. Shikh-Bahaei, "Efficient resource allocation for mobile-edge computing networks with NOMA: Completion time and energy minimization," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7771–7784, Nov. 2019.

[44] H. Li, T. Wei, A. Ren, Q. Zhu, and Y. Wang, "Deep reinforcement learning: Framework, applications, and embedded implementations: Invited paper," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 847–854.

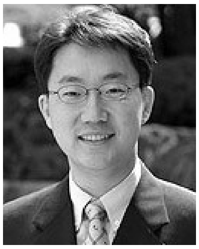
[45] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018.

[46] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

MAURICE NDUWAYEZU received the B.S. degree from University of Rwanda, Rwanda, in 2012, and the M.S. degree from Inje University, South Korea, in 2019, both in electronics and telecommunication engineering. His research interest includes the application of machine learning in wireless communication networks optimization and big data analytics.

QUOC-VIET PHAM (Member, IEEE) received the B.S. degree in electronics and telecommunications engineering from the Hanoi University of Science and Technology, Vietnam, in 2013, and the M.S. and Ph.D. degrees in telecommunications engineering from Inje University, South Korea, in 2015 and 2017, respectively. He is currently a Research Professor at the Research Institute of Computer, Information and Communication, Pusan National University, South Korea. From September 2017 to December 2019, he was with Kyung Hee University, Changwon National University, and Inje University on various academic positions. He received the Best Ph.D. Thesis Award in Engineering from Inje University, in 2017. His research interests include convex optimization, game theory, and machine learning to mobile edge/cloud computing and resource allocation for 5G wireless networks and beyond.



WON-JOO HWANG (Senior Member, IEEE) received the B.S. and M.S. degrees in computer engineering from Pusan National University, Pusan, South Korea, in 1998 and 2000, respectively, and the Ph.D. degree in information systems engineering from Osaka University, Osaka, Japan, in 2002. From 2012 to 2019, he was a Full Professor at Inje University, Gimhae, South Korea. Since January 2020, he has been a Full Professor with the School of Biomedical Convergence Engineering, Pusan National University, Yangsan, South Korea. His research interests include network optimization and cross layer design.