Joint Computation Offloading and Resource Allocation for D2D-Assisted Mobile Edge Computing

Wei Jiang, Daquan Feng, *Member, IEEE,* Yao Sun, *Senior Member, IEEE,*
Gang Feng, *Senior Member, IEEE,* Zhenzhong Wang, and Xiang-Gen Xia, *Fellow, IEEE*

Abstract—Computation offloading via device-to-device communications can improve the performance of mobile edge computing by exploiting the computing resources of user devices. However, most proposed optimization-based computation offloading schemes lack self-adaptive abilities in dynamic environments due to time-varying wireless environment, continuous-discrete mixed actions, and coordination among devices. The conventional reinforcement learning based approaches are not effective for solving an optimal sequential decision problem with continuous-discrete mixed actions. In this paper, we propose a hierarchical deep reinforcement learning (HDRL) framework to solve the joint computation offloading and resource allocation problem. The proposed HDRL framework has a hierarchical actor-critic architecture with a meta critic, multiple basic critics and actors. Specifically, a combination of deep Q-network (DQN) and deep deterministic policy gradient (DDPG) is exploited to cope with the continuous-discrete mixed action spaces. Furthermore, to handle the coordination among devices, the meta critic acts as a DQN to output the joint discrete action of all devices and each basic critic acts as the critic part of DDPG to evaluate the output of the corresponding actor. Simulation results show that the proposed HDRL algorithm can significantly reduce the task computation latency compared with baseline offloading schemes.

Index Terms—Computation offloading, resource allocation, mobile edge computing, device-to-device, deep reinforcement learning.

1 Introduction

The development of smart mobile devices (MDs) promotes the booming of new applications, such as natural language processing, face recognition, interactive gaming, virtual/augment reality, e-health and e-education. According to Grand View Research, the global mobile application market size is expected to grow at a compound annual growth rate of 11.5% from 2020 to 2027 [1]. These emerging applications usually demand extensive computing capability, vast battery energy, and short response time. However, the computing capability and battery level of MDs are in general constrained. Executing computation-intensive applications or tasks locally at MDs poses severe challenges to users' quality of experience (QoE).

To address this issue, mobile cloud computing (MCC), which allows MDs offload the computation tasks to remote clouds for processing, has been intensively investigated [2], [3], [4]. However, the long transmission distances from MDs to the remote cloud usually cause high transmission latency of MDs. To address this problem, mobile edge computing (MEC) has emerged as a promising paradigm to provide cloud-computing capabilities within radio access networks (RANs) [5], [6], [7]. By offloading the computation tasks to MEC servers in proximity to MDs, the computing performance, including the latency and energy consumption, could be greatly improved.

Meanwhile, with the device-to-device (D2D) communication mode specified by the Third Generation Partnership Project (3GPP) [8], offloading the computation tasks to nearby MDs via D2D communication links has been considered as an efficient solution to support computing intensive applications [9]. This computation offloading mode can effectively reduce the transmission latency and energy consumption of MDs. By taking the advantages of the proximity, spatial reuse, traffic offloading gain, and better coverage of D2D communications, D2D-assisted MEC offloading architectures can help mobile users enjoy ubiquitous edge computing anywhere and anytime. However, most current optimization or game based methods [10], [11], [12] require the prior information of environment statistics and cannot be efficiently applied to dynamic D2D-assisted MEC offloading systems.

In fact, deploying D2D-assisted MEC offloading service faces several challenging issues. First, various computation nodes (*i.e.*, MEC servers and MDs) have different network conditions and resource constraints, and it is necessary to design efficient offloading policies while considering the

- *This work was supported in part by the National Science and Technology Major Project under Grant 2020YFB1807601, and the Shenzhen Science, and Technology Program under Grants JCYJ20210324095209025, and in part by ZTE Industry-Academia-Research Cooperation Funds. D. Feng is the corresponding author (e-mail: fdquan@gmail.com).*
- *W. Jiang and D. Feng are with the Shenzhen Key Laboratory of Digital Creative Technology, the Guangdong Province Engineering Laboratory for Digital Creative Technology, the Guangdong-Hong Kong Joint Laboratory for Big Data Imaging and Communication, College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China.*
- *Y. Sun is with James Watt School of Engineering, University of Glasgow, G12 8QQ, Scotland, UK.*
- *G. Feng is with the Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou 313001, China, and also with the National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu 611731, China.*
- *Z. Wang is with the Technical Management Center, China Media Group, Beijing 100020, China.*
- *X.-G. Xia is with the Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716, USA.*

complex and dynamic network and resource states. Second, some computation tasks can be partitioned for local and remote executions. Partial computation offloading mode makes offloading policy design even more complicated. Third, computation offloading at RANs requires abundant spectrum resource for data transmission. Spectrum reuse may result in interference which may further increase the transmission latency, *e.g.*, due to Automatic Repeat Request (ARQ). Hence, radio spectrum resource allocation is critical for improving users' QoE in a spectrum shared D2D-assisted MEC offloading system. Last, intuitively time-varying wireless channel condition may greatly affect the optimal offloading decision in a D2D-assisted MEC system. Thus, it is non-trivial to design an optimal offloading policy which can robustly and dynamically adapt to time-varying wireless environment.

To address the afore-mentioned challenges, in this paper, we investigate joint computation offloading and resource allocation problem for D2D-assisted MEC networks with aim to minimize the total task computation latency. We consider a multi-cell, multi-server, multi-user heterogeneous network where each MEC server is co-located with a base station (BS). An MD can partially offload its computation task to either an MEC server or a nearby MD via a cellular or a D2D link, respectively. The joint computation offloading and resource allocation problem is formulated as a mixed-integer and non-convex programming problem and the difficulty of solving the problem lies in three aspects: time-varying wireless environment, continuous and discrete mixed action spaces, and coordination among MDs. To handle this problem, we propose a hierarchical deep reinforcement learning (HDRL) framework with a hierarchical actor-critic architecture. The proposed HDRL framework can automatically learn the network environment and generate the continuous action (offloading ratio) and discrete actions (offloading node and sub-channel selection) to minimize the total task computation latency. Simulation results show that the proposed HDRL framework can significantly reduce the average task computation latency when compared with other popular offloading schemes. The main contributions of this paper are summarized as follows.

1) *Combination of DQN and DDPG:* A combination of DQN and DDPG is exploited to cope with the continuous-discrete mixed action space. It utilizes actor to output continuous action, while uses critic to evaluate the action of actor and output the discrete action. The continuous action and discrete action can be jointly obtained without discretizing or relaxing the action space.
2) *Hierarchical Actor-Critic Architecture:* To learn continuous actions and joint discrete actions simultaneously, we propose a hierarchical actor-critic architecture with two level hierarchy. At the first level of hierarchy, each basic critic acts as the critic part of DDPG to evaluate the actions of the corresponding actor. At the second level of hierarchy, the meta critic acts as a DQN to output the joint discrete action of all devices, to realize the coordination among devices.
3) *Adaptive Offloading Method:* The proposed HDRL algorithm is an online learning method which is free of offline training phase and updates deep neural networks (DNNs) online. The proposed method can robustly and dynamically adapt to time-varying wireless environment.
4) *Higher Computation Offloading Performance:* Simulation results show that our proposed HDRL based algorithm achieves 12% performance improvement on average task computation latency compared with DQN based offloading scheme.

The remainder of this paper is organized as follows. The related work is surveyed in Section 2. In Section 3, we present the system model and formulate the joint computation offloading and resource allocation problem. An HDRL framework is proposed to solve the problem in Section 4. In Section 5, we evaluate the performance of the proposed algorithm by simulation. Finally, we conclude this paper in Section 6.

2 Related Work

The design of computation offloading policies has been intensively investigated in recent years. Most studies exploited optimization or game theory methods to solve the computation offloading problems [13], [14], [15], [16], [17], [18], [19], [20], [21]. Considering binary offloading where a task has to be executed locally or offloaded to the edge server as a whole, a Lyapunov optimization-based dynamic computation offloading algorithm was proposed in [13] to jointly decide the offloading decision, CPU-cycle frequencies, and transmit power. Under the assumption that a task can be arbitrarily partitioned for parallel processing, the authors of [14] exploited partial offloading and proposed a locally optimal algorithm with the univariate search technique to minimize the execution latency and energy consumption. They demonstrated that partial offloading can achieve lower execution latency and energy consumption when compared with binary offloading. Chen *et al.* [15] studied the multi-user computation offloading problem in a multi-channel wireless interference environment and proposed a game theoretic approach for achieving efficient computation offloading in a distributed manner. Zhang *et al.* [16] investigated the energy-efficient resource allocation problem for cloud services and proposed a two-phase framework to minimize the energy consumption, where a multi-threshold-based classification scheme was used to classify various hosts into different groups in the first phase, and then a metaheuristic search method was developed to search an energy-efficient host for allocating its resource to different services in the second phase.

To maximize the utilization of the computation resources in both MEC servers and mobile devices, He *et al.* [17] proposed an optimal task offloading policy to maximize the system computation capacity for a multi-user D2D-enabled MEC system. A computation offloading scheme based on alternating direction method was proposed in [18] to minimize the computation latency. To reduce the implementation complexity, a heuristic scheme based on greedy task assignment was developed in [19]. Considering the interference between cellular and D2D links with shared spectrum, a joint partial offloading and resource allocation scheme was

proposed in [20] to minimize the overall computation latency. With the involvement of vehicular computing and fog computing, Yadav *et al.* [21] proposed a heuristic approach based offloading policy to minimize energy consumption and service latency in a vehicular fog environment. However, these optimization or game based solutions require the prior information of environment statistics and cannot be efficiently applied to dynamic D2D-enabled MEC systems.

Some studies have applied machine learning techniques to solve the computation offloading problems [22], [23], [24], [25], [26], [27]. Addressing the issue of time-varying channel state information (CSI) in MEC systems, a reinforcement learning (RL) based offloading policy was developed in [22] to maximize users' long-term utilities. To speed up decision-making time, the authors of [23] proposed an RL based computation offloading scheme to minimize latency and energy consumption for edge-enable sensor networks. Considering random user arrivals, Huang *et al.* [24] proposed to integrate RL and stochastic gradient descent to improve system performance in an online manner. To handle the high dimensionality in state space, the authors of [25] used a DQN to learn the optimal offloading policy in which deep neural network is used to approximate the Q-function. A double DQN (DDQN) based algorithm was developed in [26] to solve the computation offloading problem without knowing a priori knowledge of network dynamics. In [27], the authors investigated the joint resource allocation and task scheduling problem and proposed a deep RL (DRL) based offloading approach which adopts the actor-critic architecture to generate continuous action and then maps the action to a discrete one to handle the large action space. However, these studies only deal with the computation offloading problems with either a discrete action space or a continuous action space. In reality, the action space of the computation offloading problem is often discrete-continuous mixed, *i.e.*, discrete and continuous actions need to be jointly decided to accomplish the offloading process. For example, we should not only select the offloading node or the transmission channel, but also decide the offloading ratio to minimize the computation latency.

The authors of [28] proposed a multi-task learning method to jointly optimize the offloading decision and computational resource allocation. However, this method is based on offline training and requires training datasets. Since appropriate training datasets may not be available at hand, it is more suitable to use an online learning method to make the offloading decision over the time. A hybrid decision controlled actor-critic learning method was proposed in [29] for the dynamic offloading problem, where the actor uses deep deterministic policy network to output continuous action and the critic uses deep Q-network to output discrete action and also evaluate the actor's output. However, the authors ignored the transmission interference by allocating orthogonal channels to different users in an MEC system. In general, D2D communication operating in underlay mode shares the cellular spectrum by frequency reuse. With the involved D2D communications, the coordination among devices need to be addressed to satisfy the constraints in offloading node selection and spectrum reuse. Therefore, an efficient learning based method is needed to solve the joint computation offloading and resource allocation problem with time-varying wireless environment, continuous-discrete mixed actions, and coordination among devices for a D2D-enabled MEC system.

3 System Model and Problem Formulation

In this section, we first present the network model and then describe the task, computation and communication models in details. Thereafter, we formulate the joint computation offloading and resource allocation problem with the aim of minimizing the total task computation latency.

3.1 Network Model

We consider a multi-cell, multi-server, multi-user heterogeneous network with D2D communications, as illustrated in Fig. 1. There are M BSs deployed in the network and an MEC server is deployed with each BS which provides computation offloading service to the mobile devices [30]. In general, an MEC server can be either a physical server or a virtual machine with a certain computation capability and can communicate with the mobile devices via cellular links provided by the co-located BS [31]. The set of MEC servers is denoted as $\mathcal{M} = \{1, 2, ..., M\}$. For convenience of expression, we will refer to the BS set and MEC server set interchangeably. There are two types of mobile devices, namely, request devices (RDs) and service devices (SDs) [32]. Let $\mathcal{U} = \{1, 2, ..., U\}$ denote the set of RDs which have a computation task in a time slot t, $t \in \mathcal{T} = \{1, 2, ..., T\}$, where T is the finite time horizon. The set $\mathcal{V} = \{1, 2, ..., V\}$ of SDs consists of all idle mobile devices which can offer computation offloading services for RDs.

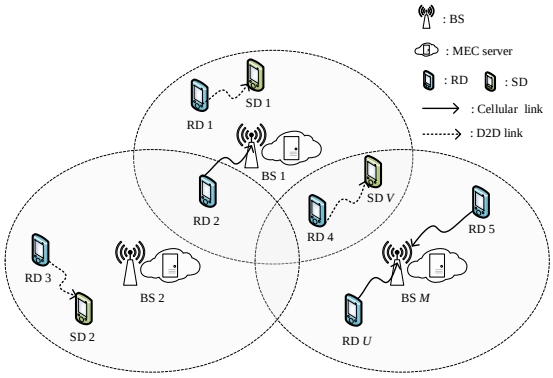


Fig. 1. The system model.

Let $L_{BS,m} \in \mathbb{R}^2$, $L_{RD,u} \in \mathbb{R}^2$ and $L_{SD,v} \in \mathbb{R}^2$ denote the locations of BS m, $m \in \mathcal{M}$, RD u, $u \in \mathcal{U}$ and SD v, $v \in \mathcal{V}$, respectively. We adopt a simple communication protocol to determine the coverage area of BSs and the neighbouring SDs for D2D communications [33]. Essentially, RD u is in the coverage area of BS m if $\|L_{BS,m} - L_{RD,u}\| < R_{BS}$, where R_{BS} is the cell coverage radius. Note that each RD can be located in the coverage area of one or more BSs. For example, in Fig. 1, RD 2, which lies in the overlapping coverage areas of BS 1 and BS 2, can communicate with either one of these BSs. Moreover, SD v is a neighbour of RD u if $\|L_{SD,v} - L_{RD,u}\| < R_{D2D}$, where R_{D2D} is the maximum distance of D2D communication.

Each RD can offload its computation task to either an MEC server or a neighbouring SD via cellular or D2D link respectively. The set of MEC servers that RD u can communicate with is defined as $\mathcal{N}_{MEC,u} = \{m \,|m \in \mathcal{M}, \|L_{BS,m} - L_{RD,u}\| < R_{BS}\}$. Moreover, the set of neighbouring SDs of RD u is denoted by $\mathcal{N}_{D2D,u} = \{v \,|v \in \mathcal{V}, \|L_{SD,v} - L_{RD,u}\| < R_{D2D}\}$. We assume that an SD can serve at most one offloading RD at a time, resulting in non-overlapping D2D pairs in the network. Note that the network can stimulate and reward the SDs to serve others and benefits from that [9]. We focus on the resource management in this paper and leave the incentive mechanism for future work. For better readability, we list the key notations used in this paper in Table 1.

TABLE 1
Notations

Notation	Definition
$\mathcal{M}$	Set of MEC servers
$\mathcal{U}$	Set of RDs
$\mathcal{V}$	Set of SDs
$\mathcal{K}$	Set of sub-channels
b_u^t	Task input data size
c_u^t	Required number of CPU cycles per bit
α_u^t	Offloading ratio
$f_{\text{RD},u}^t$	Computation capacity of RD u
$f_{SD,v}^t$	Computation capacity of SD v
$f_{MEC,m}^t$	Computation capacity of MEC server m
$x_{u,v}^t$	D2D offloading indicator
$X_{u,m}^t$	MEC offloading indicator
$y_{u,v,k}^t$	Sub-channel assignment indicator for D2D link
$Y_{u,m,k}^t$	Sub-channel assignment indicator for cellular link
p_u^t	Transmit power
$H_{u,m,k}^t/h_{u,v,k}^t$	Channel gain of cellular/D2D link
$W_{u,m,k}^t/w_{u,v,k}^t$	Transmission rate of cellular/D2D link
$E_{RD,u}^t$	Energy consumption of RD u
$E_{SD,v}^t$	Energy consumption of SD v
L_u^t	Task computation latency

3.2 Task Model

Let us denote the computation task generated by RD u in time slot t as tuple $J_u^t = (b_u^t, c_u^t)$, where b_u^t and c_u^t denote the size of input data (in bits) and the number of CPU cycles required for computing one-bit data respectively. In this paper, we consider data-partition applications, where the input data can be arbitrarily partitioned for parallel processing due to bit-wise independence. Such kind of applications include virus scan, image compression and recognition applications, *etc.* [34]. This implies that a fraction of this kind of task can be processed locally, and the rest can be offloaded either to an MEC server or a neighbouring SD for parallel execution.

We define the offloading ratio as $\alpha_u^t \in [0, 1]$ for task J_u^t, which can be considered as the percentage of the input data size (in bit) to be offloaded. Specifically, $(1 - \alpha_u^t) \cdot b_u^t$ bits are processed locally, while $\alpha_u^t \cdot b_u^t$ bits are offloaded for remote execution. Hence, the computation of a task involves local execution, remote execution and communication, which are discussed in the following in detail.

3.3 Computation Model

1) *Local Execution*: Let $f_{\text{RD},u}^t$ (in cycles/s) denote the available computational capacity of RD u, $u \in \mathcal{U}$ in time slot t. Thus, the task execution time for processing a fraction of task J_u^t locally in RD u can be derived as

$$L_{loc,exe}^{u,t} = \frac{(1 - \alpha_u^t) \cdot b_u^t \cdot c_u^t}{f_{\text{RD},u}^t}, \tag{1}$$

where $(1 - \alpha_u^t) \cdot b_u^t \cdot c_u^t$ is the total CPU cycles required for computing $(1 - \alpha_u^t)$ of task J_u^t. The energy consumption of RD u for processing $(1 - \alpha_u^t)$ of task J_u^t locally is given as

$$E_{loc,exe}^{u,t} = \kappa_0 \cdot (f_{\text{RD},u}^t)^2 \cdot (1 - \alpha_u^t) \cdot b_u^t \cdot c_u^t, \tag{2}$$

where κ_0 is the energy coefficient depending on the chip architecture [18], [20] and set as 10^{-27} [18].

2) *D2D Execution*: Let $f_{\text{SD},v}^t$ (in cycles/s) denote the available computational capacity of SD v, $v \in \mathcal{V}$ in time slot t. We use D2D offloading binary variable $x_{u,v}^t \in \{0, 1\}$ to indicate whether RD u, $u \in \mathcal{U}$ offloads its task to neighbouring SD v, $v \in \mathcal{N}_{D2D,u}$ or not: $x_{u,v}^t = 1$ if RD u offloads its task to SD v and 0 otherwise. We assume that SD v can serve at most one offloading RD at a single time slot, and thus we have $\sum_{u \in \mathcal{U}} x_{u,v}^t \leq 1$. The task execution time of processing α_u^t of task J_u^t in neighbouring SD v can be calculated as

$$L_{D2D,exe}^{u,t} = \sum_{v \in \mathcal{N}_{D2D,u}} \frac{x_{u,v}^t \cdot \alpha_u^t \cdot b_u^t \cdot c_u^t}{f_{\text{SD},v}^t}. \tag{3}$$

The energy consumption of SD v for processing α_u^t of task J_u^t can be calculated as

$$E_{SD,v}^t = \sum_{u \in \mathcal{U}} x_{u,v}^t \cdot \kappa_0 \cdot (f_{\text{SD},v}^t)^2 \cdot \alpha_u^t \cdot b_u^t \cdot c_u^t. \tag{4}$$

3) *MEC Execution*: Let $f_{MEC,m}^t$ (in cycles/s) denote the available computational capacity of MEC server m, $m \in \mathcal{M}$ in time slot t. We use MEC offloading binary variable $X_{u,m}^t \in \{0, 1\}$ to indicate whether RD u, $u \in \mathcal{U}$ offloads its task to MEC server m, $m \in \mathcal{N}_{MEC,u}$ or not: $X_{u,m}^t = 1$ if RD u offloads its task to MEC server m and 0 otherwise. We assume that each MEC server adopts a non-preemptive CPU allocation scheme, *i.e.*, MEC server m allocates all computation resource to a task until the task is completed [35]. Each MEC server can execute the tasks based on the first-come first-served (FCFS) policy [18]. Let $q_{m,u}^t$ be the index of task J_u^t in the process sequence of MEC server m, and tasks are executed in the ascending order of $q_{m,u}^t$. The task execution time of processing α_u^t of task J_u^t in MEC server m can be calculated as

$$L_{MEC,exe}^{u,t} = \sum_{m \in \mathcal{N}_{MEC,u}} \frac{X_{u,m}^t \cdot \left[\alpha_u^t \cdot b_u^t \cdot c_u^t + \sum\limits_{\substack{i \in \mathcal{U}, \\ q_{m,i}^t < q_{m,u}^t}} X_{i,m}^t \cdot \alpha_i^t \cdot b_i^t \cdot c_i^t \right]}{f_{MEC,m}^t}. \tag{5}$$

3.4 Communication Model

We assume that the cellular and D2D communication links share the same frequency band. We adopt OFDMA scheme in the uplink [17]. The available bandwidth B_{tot} is divided into K orthogonal sub-channels. Let $\mathcal{K} = \{1, 2, ..., K\}$

denote the set of K orthogonal sub-channels, and the bandwidth of each sub-channel is B (in Hz). Since we consider a kind of data-partition applications (*e.g.*, the face detection and recognition application) which usually have similar input data sizes, it is reasonable to assume that each RD is assigned to one sub-channel [12], [17], [36]. In addition, to ensure the orthogonality of uplink transmissions among RDs in the same cell, we assume that RDs in the same cell are allocated with different sub-channels, and only the suffered interferences from other cells and D2D links are considered [18]. Note that the communication resource allocation mainly involves the allocation of available sub-channels. The appropriate sub-channel allocation can decrease the mutual interference and transmission latency.

We define sub-channel assignment binary variable $Y^t_{u,m,k} \in \{0,1\}$ to indicate whether RD u, $u \in \mathcal{U}$ linked to MEC server m, $m \in \mathcal{N}_{MEC,u}$ is assigned sub-channel k, $k \in \mathcal{K}$ in time slot t: $Y^t_{u,m,k} = 1$ if RD u linked to MEC server m is assigned sub-channel k and 0 otherwise. Let binary variable $y^t_{u,v,k} \in \{0,1\}$ indicate whether RD u, $u \in \mathcal{U}$ linked to neighbouring SD v, $v \in \mathcal{N}_{D2D,u}$ is assigned sub-channel k, $k \in \mathcal{K}$ in time slot t: $y^t_{u,v,k} = 1$ if RD u linked to neighbouring SD v is assigned sub-channel k and 0 otherwise. To make effective use of the spectrum, we assume a sub-channel can be shared by at most two RDs simultaneously [20].

Let $H^t_{u,m,k}$ and $h^t_{u,v,k}$ denote the channel gain of the communication links from RD u, $u \in \mathcal{U}$ to MEC server m, $m \in \mathcal{M}$ and SD v, $v \in \mathcal{V}$ on sub-channel k, $k \in \mathcal{K}$ in time slot t, respectively. The channel gains incorporate independent Rayleigh fading and distance based path loss [20], which can be expressed as, respectively

$$H^t_{u,m,k} = G^t_{u,m,k} \cdot D^{-\varsigma}_{u,m}, \tag{6}$$

and

$$h^t_{u,v,k} = g^t_{u,v,k} \cdot d^{-\varsigma}_{u,v}, \tag{7}$$

where $G^t_{u,m,k}$ and $g^t_{u,v,k}$ denote independent Rayleigh fading; $D^{-\varsigma}_{u,m}$ and $d^{-\varsigma}_{u,v}$ are path loss. Here, $D_{u,m}$ and $d_{u,v}$ are the distances from RD u, $u \in \mathcal{U}$ to MEC server m, $m \in \mathcal{M}$ and SD v, $v \in \mathcal{V}$, respectively; and ς is the path loss exponent.

Let p^t_u denote the transmit power of RD u, $u \in \mathcal{U}$. Then, the transmission rates $W^t_{u,m,k}$ and $w^t_{u,v,k}$ from RD u, $u \in \mathcal{U}$ to MEC server m, $m \in \mathcal{N}_{MEC,u}$ and neighbouring SD v, $v \in \mathcal{N}_{D2D,u}$ on sub-channel k, $k \in \mathcal{K}$ in time slot t can be derived using Shannon formula as, respectively

$$W^t_{u,m,k} = B \cdot \log_2(1 + SINR^{u,m,k,t}_{MEC}), \tag{8}$$

and

$$w^t_{u,v,k} = B \cdot \log_2(1 + SINR^{u,v,k,t}_{D2D}), \tag{9}$$

where $SINR^{u,m,k,t}_{MEC}$ and $SINR^{u,v,k,t}_{D2D}$ are the signal to interference plus noise ratios (SINRs) at MEC and SD, which can be expressed as, respectively

$$SINR^{u,m,k,t}_{MEC} = \frac{p^t_u \cdot H^t_{u,m,k}}{\sigma^2 + \sum\limits_{\substack{i \in \mathcal{U} \\ i \neq u}} p^t_i \cdot H^t_{i,m,k} \cdot (\sum\limits_{j \in \mathcal{N}_{D2D,i}} y^t_{i,j,k} + \sum\limits_{n \in \mathcal{N}_{MEC,i}} Y^t_{i,n,k})}, \tag{10}$$

and

$$SINR^{u,v,k,t}_{D2D} = \frac{p^t_u \cdot h^t_{u,v,k}}{\sigma^2 + \sum\limits_{\substack{i \in \mathcal{U} \\ i \neq u}} p^t_i \cdot h^t_{i,v,k} \cdot (\sum\limits_{j \in \mathcal{N}_{D2D,i}} y^t_{i,j,k} + \sum\limits_{n \in \mathcal{N}_{MEC,i}} Y^t_{i,n,k})}, \tag{11}$$

where σ^2 is the additive white Gaussian noise (AWGN).

Therefore, the transmission time of RD u for offloading α^t_u of task J^t_u to MEC server m and neighbouring SD v can be calculated as, respectively

$$L^{u,t}_{MEC,trans} = \sum_{m \in \mathcal{N}_{MEC,u}} \sum_{k \in \mathcal{K}} \frac{Y^t_{u,m,k} \cdot \alpha^t_u \cdot b^t_u}{W^t_{u,m,k}}, \tag{12}$$

and

$$L^{u,t}_{D2D,trans} = \sum_{v \in \mathcal{N}_{D2D,u}} \sum_{k \in \mathcal{K}} \frac{y^t_{u,v,k} \cdot \alpha^t_u \cdot b^t_u}{w^t_{u,v,k}}. \tag{13}$$

The energy consumption of RD u for offloading α^t_u of task J^t_u to MEC server m and neighbouring SD v can be calculated as, respectively

$$E^{u,t}_{MEC,trans} = \sum_{m \in \mathcal{N}_{MEC,u}} \sum_{k \in \mathcal{K}} \frac{Y^t_{u,m,k} \cdot p^t_u \cdot \alpha^t_u \cdot b^t_u}{W^t_{u,m,k}}, \tag{14}$$

and

$$E^{u,t}_{D2D,trans} = \sum_{v \in \mathcal{N}_{D2D,u}} \sum_{k \in \mathcal{K}} \frac{y^t_{u,v,k} \cdot p^t_u \cdot \alpha^t_u \cdot b^t_u}{w^t_{u,v,k}}. \tag{15}$$

Note that only the latency of uplink transmission is considered, and the latency of computation outcome transmission from the MEC server or neighbouring SD to RD u is neglected in this paper. This is because the size of computation outcome data in general is much smaller than that of the computation input data including the system settings, input parameters and program codes [37].

3.5 Problem Formulation

Since each task can be processed locally and offloaded for remote execution concurrently, the computation latency of task J^t_u is determined by the maximal value of $L^{u,t}_{loc,exe}$, $L^{u,t}_{D2D,trans} + L^{u,t}_{D2D,exe}$ and $L^{u,t}_{MEC,trans} + L^{u,t}_{MEC,exe}$. Thus, the computation latency of task J^t_u can be written as

$$\begin{aligned} L^t_u = max\{&L^{u,t}_{loc,exe}, L^{u,t}_{D2D,trans} + L^{u,t}_{D2D,exe}, \\ &L^{u,t}_{MEC,trans} + L^{u,t}_{MEC,exe}\}. \end{aligned} \tag{16}$$

In addition, the energy consumption of RD u for task J^t_u includes the energy consumed by local execution and data transmission. Therefore, the energy consumption of RD u for task J^t_u can be calculated as

$$E^t_{RD,u} = E^{u,t}_{loc,exe} + E^{u,t}_{D2D,trans} + E^{u,t}_{MEC,trans}. \tag{17}$$

The objective of the joint computation offloading and resource allocation problem is to minimize the long-term

total task computation latency, which can be formulated as follows:

$$\min_{\alpha_u^t, x_{u,v}^t, X_{u,m}^t, y_{u,v,k}^t, Y_{u,m,k}^t} \sum_{t\in\mathcal{T}}\sum_{u\in\mathcal{U}} L_u^t, \tag{18}$$

$$s.t.\ \alpha_u^t \in [0,1], u\in\mathcal{U}, t\in\mathcal{T}, \tag{18a}$$

$$\sum_{v\in\mathcal{N}_{D2D,u}} x_{u,v}^t + \sum_{m\in\mathcal{N}_{MEC,u}} X_{u,m}^t = \lceil \alpha_u^t \rceil, u\in\mathcal{U}, t\in\mathcal{T}, \tag{18b}$$

$$\sum_{u\in\mathcal{U}} x_{u,v}^t \le 1, v\in\mathcal{V}, t\in\mathcal{T}, \tag{18c}$$

$$\sum_{k\in\mathcal{K}} y_{u,v,k}^t = x_{u,v}^t, u\in\mathcal{U}, v\in\mathcal{N}_{D2D,u}, t\in\mathcal{T}, \tag{18d}$$

$$\sum_{k\in\mathcal{K}} Y_{u,m,k}^t = X_{u,m}^t, u\in\mathcal{U}, m\in\mathcal{N}_{MEC,u}, t\in\mathcal{T}, \tag{18e}$$

$$\sum_{u\in\mathcal{U}} Y_{u,m,k}^t \le 1, m\in\mathcal{M}, k\in\mathcal{K}, t\in\mathcal{T}, \tag{18f}$$

$$\sum_{u\in\mathcal{U}}\sum_{m\in\mathcal{N}_{MEC,u}} Y_{u,m,k}^t + \sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{N}_{D2D,u}} y_{u,v,k}^t \le 2, k\in\mathcal{K}, t\in\mathcal{T}, \tag{18g}$$

$$E_{RD,u}^t \le E_{RD,u}^{\max}, u\in\mathcal{U}, t\in\mathcal{T}, \tag{18h}$$

$$E_{SD,v}^t \le E_{SD,v}^{\max}, v\in\mathcal{V}, t\in\mathcal{T}, \tag{18i}$$

$$x_{u,v}^t, X_{u,m}^t \in \{0,1\}, u\in\mathcal{U}, v\in\mathcal{N}_{D2D,u}, m\in\mathcal{N}_{MEC,u}, t\in\mathcal{T}, \tag{18j}$$

$$y_{u,v,k}^t, Y_{u,m,k}^t \in \{0,1\}, u\in\mathcal{U}, v\in\mathcal{N}_{D2D,u}, m\in\mathcal{N}_{MEC,u}, k\in\mathcal{K}, t\in\mathcal{T}. \tag{18k}$$

In (18b), $\lceil . \rceil$ represents the ceiling function, and (18b) indicates that each RD can only choose one of the computation offloading nodes for remote execution when the offloading ratio $\alpha_u^t \neq 0$. Inequality (18c) indicates that SD v can serve at most one offloading RD at a specific time slot. (18d) and (18e) indicate that once an RD chooses a node to offload its computation task, a sub-channel should be assigned to the RD. (18f) and (18g) are the sub-channel reuse constraints, where (18f) indicates that a sub-channel can only be assigned to one cellular link in the same cell and (18g) indicates that a sub-channel can be shared by at most two links. (18h) and (18i) indicate that the energy consumption of RDs and SDs are restricted, where $E_{RD,u}^{\max}$ and $E_{SD,v}^{\max}$ are the maximum energies provided by RD u and SD v in each time slot, respectively.

It can be seen that problem (18) is a mixed-integer and non-convex programming problem, and the difficulty of solving the problem lies in many aspects. First, it involves both the continuous variable α_u^t and 0-1 integer variables $x_{u,v}^t, X_{u,m}^t, y_{u,v,k}^t, Y_{u,m,k}^t$. Second, we consider the realistic scenarios, where the wireless channel conditions, the computation resources of mobile devices and MEC servers are changing dynamically. Third, with the constraints in offloading node selection and spectrum reuse, the decisions of all RDs on offloading node selection and sub-channel selection need to be jointly addressed. Therefore, it is difficult or infeasible to find the optimal solution by employing conventional optimization-based methods. Thus, in this paper, we resort to the reinforcement learning (RL) techniques to solve this problem.

4 Hierarchical Deep Reinforcement Learning based Computation Offloading and Resource Allocation

Although the RL techniques have been widely used in solving the dynamic offloading problems, most studies deal with either a discrete action space or a continuous action space, which are not suitable for the discrete and continuous mixed action spaces. To cope with this, a hybrid decision controlled actor-critic learning method [29] is exploited, where the actor uses deep deterministic policy network to output continuous action and the critic uses deep Q-network to output discrete action and also evaluate the actor's output. However, due to the constraints in offloading node selection and spectrum reuse, the joint discrete actions of all RDs need to be learned to make the global decisions on offloading node selection and sub-channel selection. Therefore, a hierarchical actor-critic architecture [38], which additionally uses a meta critic to learn the joint discrete actions in a centralized way, is employed to design our hierarchical deep reinforcement learning (HDRL) algorithm for solving the joint computation offloading and resource allocation problem. In the following, we first give the definitions of the state space, action space and reward function for the problem. Then, the HDRL algorithm for computation offloading and resource allocation is presented in detail.

4.1 State, Action, and Reward Definition

There are three key elements in RL: 1) the state space, 2) the action space, and 3) the reward function. The definitions of the state space, action space and reward function for the proposed problem are given as follows.

1) State space: At the beginning of time slot t, the state includes the computation tasks J_u^t of RDs, the computation capacities $f_{RD,u}^t$, $f_{SD,v}^t$ and $f_{MEC,m}^t$ of RDs, SDs and MEC servers, respectively, and the channel gains $H_{u,m,k}^t$ and $h_{u,v,k}^t$ of cellular and D2D links respectively. For notational convenience, we denote $\boldsymbol{J}^t$, $\boldsymbol{f}_{RD}^t$, $\boldsymbol{f}_{SD}^t$, $\boldsymbol{f}_{MEC}^t$, $\boldsymbol{H}^t$ and $\boldsymbol{h}^t$ as the vectors of the corresponding states:

$$\boldsymbol{J}^t = [J_1^t, J_2^t, ..., J_U^t], \tag{19a}$$

$$\boldsymbol{f}_{RD}^t = [f_{RD,1}^t, f_{RD,2}^t, ..., f_{RD,U}^t], \tag{19b}$$

$$\boldsymbol{f}_{SD}^t = [f_{SD,1}^t, f_{SD,2}^t, ..., f_{SD,V}^t], \tag{19c}$$

$$\boldsymbol{f}_{MEC}^t = [f_{MEC,1}^t, f_{MEC,2}^t, ..., f_{MEC,M}^t], \tag{19d}$$

$$\boldsymbol{H}^t = [H_{1,1,1}^t, H_{1,1,2}^t, ..., H_{U,M,K}^t], \tag{19e}$$

$$\boldsymbol{h}^t = [h_{1,1,1}^t, h_{1,1,2}^t, ..., h_{U,V,K}^t]. \tag{19f}$$

Thus, the state $\boldsymbol{s}^t$ at time slot t can be defined as:

$$\boldsymbol{s}^t = [\boldsymbol{J}^t, \boldsymbol{f}_{RD}^t, \boldsymbol{f}_{SD}^t, \boldsymbol{f}_{MEC}^t, \boldsymbol{H}^t, \boldsymbol{h}^t] \in \mathcal{S}. \tag{20}$$

2) Action space: The action of each RD consists of three parts, namely, offloading ratio α_u^t, offloading node selection and sub-channel selection. We define N_u^t as the action of offloading node selection, where $N_u^t \in \mathcal{N}_u \triangleq \{m \,|m\in\mathcal{N}_{MEC,u}\} \cup \{M+v \,|v\in\mathcal{N}_{D2D,u}\}$. Note that $N_u^t \neq N_i^t$, $u, i \in \mathcal{U}$, if $N_u^t \in \mathcal{N}_{D2D,u}$. Then, the offloading binary variables $X_{u,m}^t$ and $x_{u,v}^t$ can be determined by

$$X_{u,m}^t = \begin{cases} 1, & N_u^t = m, \\ 0, & \text{otherwise}, \end{cases} \tag{21a}$$

$$x_{u,v}^t = \begin{cases} 1, \ N_u^t = M + v, \\ 0, \ \text{otherwise.} \end{cases} \tag{21b}$$

We also define K_u^t as the action of sub-channel selection, where $K_u^t \in \mathcal{K}$. Note that $K_u^t \neq K_i^t$, $u, i \in \mathcal{U}$, if $N_u^t = N_i^t$, and $\sum_{u=1}^{U} \mathbb{I}(K_u^t = k) \leq 2$, $k \in \mathcal{K}$, where $\mathbb{I}(\cdot)$ is the indicator function. Then, the sub-channel assignment binary variables $Y_{u,m,k}^t$ and $y_{u,v,k}^t$ can be determined by

$$Y_{u,m,k}^t = \begin{cases} 1, \ N_u^t = m, K_u^t = k, \\ 0, \ \text{otherwise,} \end{cases} \tag{22a}$$

$$y_{u,v,k}^t = \begin{cases} 1, \ N_u^t = M + v, K_u^t = k, \\ 0, \ \text{otherwise.} \end{cases} \tag{22b}$$

For known offloading node selection N_u^t and sub-channel selection K_u^t, constraints (18h) and (18i) provide feasibility conditions on α_u^t, which can be derived as

$$\alpha_u^t \geq \frac{\kappa_0 \cdot (f_{\text{RD},u}^t)^2 \cdot b_u^t \cdot c_u^t - E_{RD,u}^{\max}}{\kappa_0 \cdot (f_{\text{RD},u}^t)^2 \cdot b_u^t \cdot c_u^t - p_u^t \cdot b_u^t / W_{u,m,k}^t}, N_u^t = m, K_u^t = k, \tag{23}$$

$$\alpha_u^t \geq \frac{\kappa_0 \cdot (f_{\text{RD},u}^t)^2 \cdot b_u^t \cdot c_u^t - E_{RD,u}^{\max}}{\kappa_0 \cdot (f_{\text{RD},u}^t)^2 \cdot b_u^t \cdot c_u^t - p_u^t \cdot b_u^t / w_{u,v,k}^t}, N_u^t = M + v, K_u^t = k, \tag{24}$$

$$\alpha_u^t \leq \frac{E_{SD,v}^{\max}}{\kappa_0 \cdot (f_{\text{SD},v}^t)^2 \cdot b_u^t \cdot c_u^t}, N_u^t = M + v. \tag{25}$$

Thus, the action $\boldsymbol{a}_u^t$ of RD u performs at time slot t can be defined as:

$$\boldsymbol{a}_u^t = [\alpha_u^t, N_u^t, K_u^t] \in \mathcal{A}_u. \tag{26}$$

The joint action $\boldsymbol{a}^t$ of all RDs at time slot t is denoted as:

$$\boldsymbol{a}^t = [\boldsymbol{a}_1^t, ..., \boldsymbol{a}_U^t] \in \mathcal{A}. \tag{27}$$

3) Reward function: In general, the reward function should be related to the objective function of the proposed problem. Since the target of RL is to maximize the long-term reward and the objective of our optimization problem is to minimize the long-term task computation latency, the reward function should be negatively correlated with the task computation latency function. Hence, we define the reward function $R^t = \sum_{u \in \mathcal{U}} R_u^t$ at time slot t, where R_u^t is the reward function of a certain RD u, which is defined as

$$R_u^t = \frac{L_{local,u}^t - L_u^t}{L_{local,u}^t}. \tag{28}$$

where $L_{local,u}^t = \frac{b_u^t \cdot c_u^t}{f_{RD,u}^t}$ is the task computation latency of RD u at time slot t for totally local computing (*i.e.*, $\alpha_u^t = 0$) and L_u^t is denoted as (16).

4.2 HDRL Based Computation Offloading and Resource Allocation

In RL, the policy refers to the mapping relationship between state space and action space $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The goal of the RL agent is to find an optimal policy which maximizes the expected future rewards. In general, the future rewards are discounted by a discount factor $\gamma \in (0, 1]$, and the expected long-term discounted reward starting from state $\boldsymbol{s}$ with policy π can be expressed by the state value function:

$$V^\pi(\boldsymbol{s}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{s}^t, \boldsymbol{a}^t) | \boldsymbol{s}^0 = \boldsymbol{s}; \pi\right]. \tag{29}$$

Another important concept in RL is the state-action value function (also known as Q-function). It denotes the expected long-term discounted reward for a given state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ with policy π:

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{s}^t, \boldsymbol{a}^t) | \boldsymbol{s}^0 = \boldsymbol{s}, \boldsymbol{a}^0 = \boldsymbol{a}; \pi\right]. \tag{30}$$

The RL problem can be efficiently solved by using conventional RL methods (*e.g.*, Q-learning) when the dimensions of state space and action space are not large. However, our optimization problem is formulated in a dynamic wireless environment, which leads to a huge state space. To avoid the curse of dimensionality, deep RL (DRL), which uses deep neural network (DNN) to approximate the value function, is employed. There are two typical DRL algorithms, deep Q-network (DQN) [39] and deep deterministic policy gradient (DDPG) [40]. For discrete action spaces, DQN can be realized with two major techniques, experience replay and Q-target network. For continuous action spaces, DDPG uses an actor-critic architecture and two neural networks to approximate the policy function and value function, respectively. Each network also has its corresponding target network.

However, for an RD u, the action $\boldsymbol{a}_u = [\alpha_u, N_u, K_u]$ is composed of continuous action (*i.e.*, α_u) and discrete actions (*i.e.*,N_u and K_u), which leads to a mixed continuous-discrete action space. It is thus difficult to solve the problem directly by using DQN or DDPG. Fortunately, it can be observed that the decisions of offloading node selection N_u and sub-channel selection K_u depend on the offloading ratio α_u. Thus, by exploiting the hybrid decision controlled actor-critic learning method [29], for each RD, we can first calculate the optimal continuous action α_u for each discrete action pair (N_u, K_u), and then find the optimal discrete action pair (N_u, K_u) based on the calculated continuous actions among all discrete action pairs. However, due to the constraints in offloading node selection and spectrum reuse, the joint discrete actions of all RDs need to be learned to make the global decisions on offloading node selection and sub-channel selection. Hence, we employ a hierarchical actor-critic architecture [38], which additionally uses a meta critic to learn the joint discrete actions in a centralized way, to design our HDRL algorithm for computation offloading and resource allocation.

The hierarchical actor-critic architecture is illustrated in Fig. 2, which consists of a meta critic, multiple basic critics and actors. At the first level of hierarchy, each actor takes the state based local observation $\boldsymbol{s}_u = o_u(\boldsymbol{s})$ as input and uses a deep deterministic policy network $\mu_u(\boldsymbol{s}_u|\theta_{\mu_u})$ to output continuous actions $\boldsymbol{\alpha}_u = [\alpha_{u,N_u,K_u}]_{N_u \in \mathcal{N}_u, K_u \in \mathcal{K}}$ for all discrete action pairs. Each basic critic takes the local observation $\boldsymbol{s}_u$ and the output $\boldsymbol{\alpha}_u$ of the corresponding actor as input and uses a deep Q-network $Q_u^b(\boldsymbol{s}_u, \boldsymbol{\alpha}_u, N_u, K_u|\theta_{Q_u^b})$ to output Q-function Q_u^b for evaluating the actions of the corresponding actor. At the second level of hierarchy, the meta critic takes the state $\boldsymbol{s}$ and the outputs $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, ..., \boldsymbol{\alpha}_U]$ of all actors as input and uses a deep Q-network $Q^m(\boldsymbol{s}, \boldsymbol{\alpha}, \boldsymbol{N}, \boldsymbol{K}|\theta_{Q^m})$ to output Q-function Q^m for joint discrete action pairs $(\boldsymbol{N}, \boldsymbol{K})$, where $\boldsymbol{N} = [N_1, ..., N_U]$ and $\boldsymbol{K} = [K_1, ..., K_U]$.

Based on the hierarchical actor-critic architecture, we develop an HDRL algorithm for computation offloading and resource allocation. At time slot t, the state is $\boldsymbol{s}^t$. If RD $u \in \mathcal{U}$, the corresponding actor takes the local observation $\boldsymbol{s}_u^t$ as input and outputs continuous actions $\boldsymbol{\alpha}_u^t = [\alpha_{u,N_u,K_u}^t]_{N_u \in \mathcal{N}_u, K_u \in \mathcal{K}}$. To encourage exploration, each actor employs an exploration policy $\boldsymbol{\alpha}_u^t = \mu_u(s_u^t|\theta_{\mu_u}) + \upsilon^t$, where vector υ^t is the exploration noise which follows a normal distribution with zero mean and variance σ_υ [41]. The meta critic takes the state $\boldsymbol{s}^t$ and $\boldsymbol{\alpha}^t = [\boldsymbol{\alpha}_1^t, ..., \boldsymbol{\alpha}_U^t]$ as input and outputs Q-function Q^m for joint discrete action pairs $(\boldsymbol{N}, \boldsymbol{K})$. To achieve the trade-off between exploration and exploitation, the meta critic employs an ε-greedy

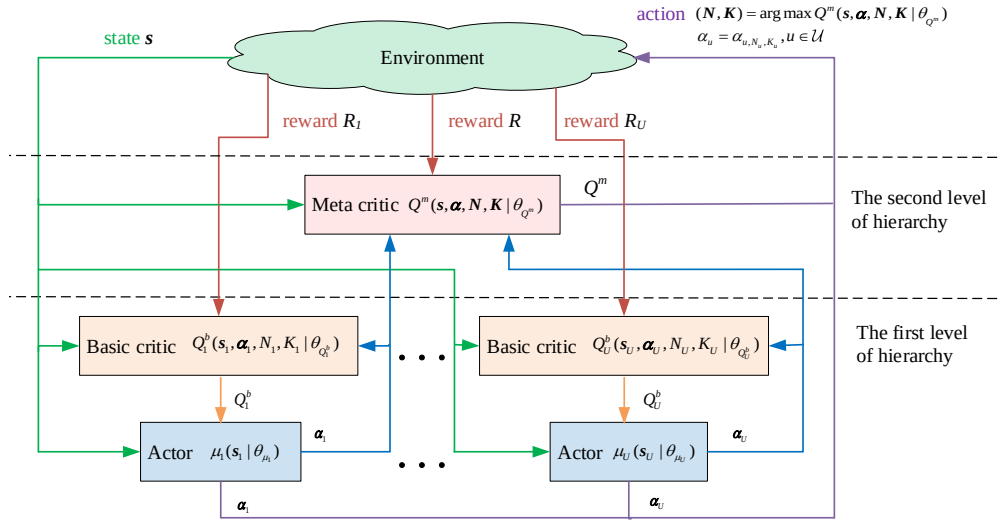


Fig. 2. Illustration of hierarchical actor-critic architecture.

policy to select the joint discrete action pair, *i.e.*, with probability ε, a random joint discrete action pair $(\boldsymbol{N}^t,\boldsymbol{K}^t)$ is selected, and otherwise the joint discrete action pair with the highest estimated Q-value $(\boldsymbol{N}^t,\boldsymbol{K}^t)=\arg\max \bar{Q}^m(\boldsymbol{s}^t,\boldsymbol{\alpha}^t,\boldsymbol{N},\boldsymbol{K}|\theta_{Q^m})$ is selected. Then, the continuous action α_u^t can be determined by $\alpha_u^t=\alpha_{u,N_u^t,K_u^t}^t$, $u\in\mathcal{U}$.

After interacting with the environment, the state transits into the next state $\boldsymbol{s}^{t+1}$ while certain reward is obtained. Then, the meta critic and basic critics store experience tuples $(\boldsymbol{s}^t,\boldsymbol{\alpha}^t,\boldsymbol{N}^t,\boldsymbol{K}^t,R^t,\boldsymbol{s}^{t+1})$ and $(\boldsymbol{s}_u^t,\boldsymbol{\alpha}_u^t,\boldsymbol{N}_u^t,\boldsymbol{K}_u^t,R_u^t,\boldsymbol{s}_u^{t+1})$, $u\in\mathcal{U}$ into their replay buffers, respectively. Random mini-batches of experience tuples are sampled from their replay buffers for training the actor and critic networks. Besides, the target actor network $\bar{\mu}_u(\boldsymbol{s}_u|\theta_{\bar{\mu}_u})$, target basic critic network $\bar{Q}_u^b(\boldsymbol{s}_u,\boldsymbol{\alpha}_u,N_u,K_u|\theta_{\bar{Q}_u^b})$ and target meta critic network $\bar{Q}^m(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{N},\boldsymbol{K}|\theta_{\bar{Q}^m})$ have the same architectures as the actor, basic critic and meta critic networks with parameters $\theta_{\bar{\mu}_u}$, $\theta_{\bar{Q}_u^b}$ and $\theta_{\bar{Q}^m}$, respectively. After calculating the target Q-values, the critic and actor networks are updated by minimizing the corresponding losses, and then the target networks are updated by tracking the learned networks.

Meta critic network updating: Since the meta critic acts as a DQN to output the joint discrete action pair with the highest Q-value, the deep Q-network $Q^m(\boldsymbol{s},\boldsymbol{\alpha},\boldsymbol{N},\boldsymbol{K}|\theta_{Q^m})$ can be iteratively updated to estimate the Q-values of joint discrete action pairs. Specifically, the meta critic network can be updated by minimizing the mean square error (MSE) loss:

$$L(\theta_{Q^m})=\frac{1}{\Omega}\sum_i\left[z_{meta}^i-Q^m(\boldsymbol{s}^i,\boldsymbol{\alpha}^i,\boldsymbol{N}^i,\boldsymbol{K}^i|\theta_{Q^m})\right]^2,\quad(31)$$

where Ω is the batch size, and

$$z_{meta}^i=R^i+\gamma\cdot\max_{\boldsymbol{N}',\boldsymbol{K}'}\bar{Q}^m(\boldsymbol{s}^{i+1},\boldsymbol{\alpha}^{i+1},\boldsymbol{N}',\boldsymbol{K}'|\theta_{\bar{Q}^m})\quad(32)$$

is the target Q-value calculated from the target meta critic network, where $\boldsymbol{\alpha}_u^{i+1}=\bar{\mu}_u(\boldsymbol{s}_u^{i+1}|\theta_{\bar{\mu}_u})$. Therefore, the gradient for updating the meta critic network can be obtained by differentiating the loss function:

$$\nabla L(\theta_{Q^m})=\frac{1}{\Omega}\sum_i(z_{meta}^i-Q^m(\boldsymbol{s}^i,\boldsymbol{\alpha}^i,\boldsymbol{N}^i,\boldsymbol{K}^i|\theta_{Q^m}))\cdot\frac{\partial Q^m(\boldsymbol{s}^i,\boldsymbol{\alpha}^i,\boldsymbol{N}^i,\boldsymbol{K}^i|\theta_{Q^m})}{\partial\theta_{Q^m}}.\quad(33)$$

Basic critic networks updating: Since each basic network acts as the critic part of DDPG to provide guidance for the training of the corresponding actor network, it can be updated by minimizing the MSE loss:

$$L(\theta_{Q_u^b})=\frac{1}{\Omega}\sum_i\left[z_{basic,u}^i-Q_u^b(\boldsymbol{s}_u^i,\boldsymbol{\alpha}_u^i,N_u^i,K_u^i|\theta_{Q_u^b})\right]^2,\quad(34)$$

where

$$z_{basic,u}^i=R_u^i+\gamma\cdot\bar{Q}_u^b(\boldsymbol{s}_u^{i+1},\boldsymbol{\alpha}_u^{i+1},N_u^{i+1},K_u^{i+1}|\theta_{\bar{Q}_u^b})\quad(35)$$

is the target Q-value calculated from each target basic critic network, where $\boldsymbol{\alpha}_u^{i+1}=\bar{\mu}_u(\boldsymbol{s}_u^{i+1}|\theta_{\bar{\mu}_u})$, and $(\boldsymbol{N}^{i+1},\boldsymbol{K}^{i+1})=\arg\max\bar{Q}^m(\boldsymbol{s}^{i+1},\boldsymbol{\alpha}^{i+1},\boldsymbol{N},\boldsymbol{K}|\theta_{\bar{Q}^m})$. The gradient for updating each basic critic network is:

$$\nabla L(\theta_{Q_u^b})=\frac{1}{\Omega}\sum_i(z_{basic,u}^i-Q_u^b(\boldsymbol{s}_u^i,\boldsymbol{\alpha}_u^i,N_u^i,K_u^i|\theta_{Q_u^b}))\cdot\frac{\partial Q_u^b(\boldsymbol{s}_u^i,\boldsymbol{\alpha}_u^i,N_u^i,K_u^i|\theta_{Q_u^b})}{\partial\theta_{Q_u^b}}.\quad(36)$$

Actor networks updating: Since each actor network acts as the actor part of DDPG to output continuous actions, it can be updated by minimizing the loss

$$L(\theta_{\mu_u})=\frac{1}{\Omega}\sum_i-Q_u^b(\boldsymbol{s}_u^i,\boldsymbol{\alpha}_u,N_u^i,K_u^i|\theta_{Q_u^b}),\quad(37)$$

where $\boldsymbol{\alpha}_u=\mu_u(\boldsymbol{s}_u^i|\theta_{\mu_u})$. The gradient for updating each actor network is:

$$\nabla L(\theta_{\mu_u})=\frac{1}{\Omega}\sum_i\frac{\partial Q_u^b(\boldsymbol{s}_u^i,\boldsymbol{\alpha}_u,N_u^i,K_u^i|\theta_{Q_u^b})}{\partial\boldsymbol{\alpha}_u}\cdot\frac{\partial\mu_u(\boldsymbol{s}_u^i|\theta_{\mu_u})}{\partial\theta_{\mu_u}}.\quad(38)$$

Target networks updating: The parameters $\theta_{\bar{Q}^m}$, $\theta_{\bar{Q}_u^b}$ and $\theta_{\bar{\mu}_u}$ of the target meta critic, target basic critic and target actor networks are updated as follows, respectively:

$$\theta_{\bar{Q}^m}\leftarrow\tau\theta_{Q^m}+(1-\tau)\theta_{\bar{Q}^m},\quad(39)$$

$$\theta_{\bar{Q}_u^b}\leftarrow\tau\theta_{Q_u^b}+(1-\tau)\theta_{\bar{Q}_u^b},\quad(40)$$

$$\theta_{\bar{\mu}_u}\leftarrow\tau\theta_{\mu_u}+(1-\tau)\theta_{\bar{\mu}_u},\quad(41)$$

with $\tau\ll 1$.

At decision time, the actors and basic critics can be run at each RD to make its own decision of offloading ratio, while the meta critic can be placed at one MEC server to jointly make the decisions of offloading node selection and sub-channel selection for each RD in a centralized way. Note that the cost of running actors and basic critics on RDs is more expensive than that on servers, and an alternative implementation scheme is to place all the actors, basic critics and meta critic at one server. This configuration can reduce the interactions between RDs and the server, while increasing the burden on the server. It needs to choose an appropriate implementation scheme according to the actual situation. The HDRL algorithm for solving the computation offloading and resource allocation problem is summarized in Algorithm 1.

The proposed HDRL algorithm has the following advantages. First, with the hierarchical actor-critic architecture, the proposed HDRL algorithm can solve the joint computation offloading and resource allocation problem with continuous-discrete mixed action space. The continuous action and discrete actions can be jointly obtained without discretizing or relaxing the action space. The proposed method can not only be applied to the system in this paper but also adapted to similar continuous-discrete mixed action based models. Second, the proposed HDRL algorithm is an online learning method which is free of offline training phase and updates DNNs online. The proposed method does not need training datasets and can robustly and dynamically adapt to time-varying wireless environment. However, the implementation of the proposed HDRL algorithm for the joint computation offloading and resource allocation problem will lead to additional computation cost in MEC servers. Note that although the computational complexity of the proposed HDRL algorithm will be higher than that of offline learning methods, it is much smaller than that of conventional optimization methods, such as the Lagrangian relaxation-based algorithms [17], [18], [19], [20].

4.3 Computation Complexity Analysis

In the following, the complexity of the proposed HDRL algorithm is analyzed. For a DNN, the time complexity mainly comes from the training process, which can be calculated as $O\left(H_i \cdot H_1 + H_1 \cdot H_2 + \cdots + H_n \cdot H_o\right)$ for a single sample, where H_i, H_o and H_n are the input, output and hidden layer sizes, respectively. Additionally, the number of times for training depends on the number of episodes, I, and time slots T. Hence, the time complexity of the actor and basic critic networks in the proposed HDRL algorithm are $O\left(I \cdot T \cdot U \cdot \Omega \cdot \left(H_i \cdot H_1 + H_1 \cdot H_2 + \cdots + H_n \cdot H_o\right)\right)$, where $I \cdot T \cdot U$is the number of times for training and Ω is the batch size. Here, the input layer size H_i depends on the state space size $|\mathcal{S}_u|$, while the output layer size H_o is determined by the action space size, *i.e.*, the number of discrete action pairs $(M+V) \cdot K$. Similarly, the time complexity of the meta critic network in the proposed HDRL algorithm is $O\left(I \cdot T \cdot \Omega \cdot \left(H_i \cdot H_1 + H_1 \cdot H_2 + \cdots + H_n \cdot H_o\right)\right)$. Here, the input layer size H_i depends on the state space size $|\mathcal{S}|$, while the output layer size H_o is determined by the action space size, *i.e.*, the number of available coordinated discrete action pairs.

5 Performance Evaluations

In this section, simulations are presented to evaluate the performance of the proposed HDRL algorithm. There are $M = 3$ BSs and a BS coverage radius is 300 m, while the BS-to-BS distance is 400 m. RDs and SDs are randomly scattered in the coverage area of the BSs. The maximum distance of D2D communication is 50 m [20]. The total bandwidth is divided into $K = 10$ orthogonal sub-channels [32]. The fading of each channel is assumed to follow Rayleigh distribution with unit variance [17]. The transmit power of RDs is 0.2 W [32]. We assume

Algorithm 1 HDRL Based Computation Offloading and Resource Allocation Algorithm

Randomly initialize actor networks $\mu_u(\cdot|\theta_{\mu_u})$, basic critic networks $Q_u^b(\cdot|\theta_{Q_u^b})$ and meta critic network $Q^m(\cdot|\theta_{Q^m})$.
Initialize target actor networks $\bar{\mu}_u(\cdot|\theta_{\bar{\mu}_u})$, target basic critic networks $\bar{Q}_u^b(\cdot|\theta_{\bar{Q}_u^b})$ and target meta critic network $\bar{Q}^m(\cdot|\theta_{\bar{Q}^m})$ with weights $\theta_{\bar{\mu}_u} = \theta_{\mu_u}$, $\theta_{\bar{Q}_u^b} = \theta_{Q_u^b}$ and $\theta_{\bar{Q}^m} = \theta_{Q^m}$.
Initialize the experience replay buffers.
for each episode **do**
  Obtain the initial state $\boldsymbol{s}^1$.
  for each time slot t **do**
    for each RD u **do**
      Get current observation $\boldsymbol{s}_u^t$.
      Obtain continuous actions $\boldsymbol{\alpha}_u^t = \mu_u(\boldsymbol{s}_u^t|\theta_{\mu_u}) + \upsilon^t$, where vector υ^t is exploration noise.
    end for
    With probability ε select a random joint discrete action pair $(\boldsymbol{N}^t, \boldsymbol{K}^t)$, and otherwise select $(\boldsymbol{N}^t, \boldsymbol{K}^t) = \arg\max Q^m(\boldsymbol{s}^t, \boldsymbol{\alpha}^t, \boldsymbol{N}, \boldsymbol{K}|\theta_{Q^m})$.
    for each RD u **do**
      Select continuous action $\alpha_u^t = \alpha_{u,N_u^t,K_u^t}^t$.
    end for
    All RDs take actions.
    Obtain reward R^t and observe the next state $\boldsymbol{s}^{t+1}$.
    Store experience tuple $(\boldsymbol{s}^t, \boldsymbol{\alpha}^t, \boldsymbol{N}^t, \boldsymbol{K}^t, R^t, \boldsymbol{s}^{t+1})$ into replay buffer $\mathcal{D}^m$.
    for each RD u **do**
      Obtain reward R_u^t and get the next observation $\boldsymbol{s}_u^{t+1}$.
      Store experience tuple $(\boldsymbol{s}_u^t, \boldsymbol{\alpha}_u^t, N_u^t, K_u^t, R_u^t, \boldsymbol{s}_u^{t+1})$ into replay buffer $\mathcal{D}_u^b$.
    end for
    Sample a batch of Ω experience tuples $(\boldsymbol{s}^i, \boldsymbol{\alpha}^i, \boldsymbol{N}^i, \boldsymbol{K}^i, R^i, \boldsymbol{s}^{i+1})$ from $\mathcal{D}^m$.
    Calculate the meta critic target Q-value z_{meta}^i according to (32).
    Update the meta critic network θ_{Q^m} by performing a gradient descent step with the gradient calculated by (33).
    for each RD u **do**
      Sample a batch of Ω experience tuples $(\boldsymbol{s}_u^i, \boldsymbol{\alpha}_u^i, N_u^i, K_u^i, R_u^i, s_u^{i+1})$ from $\mathcal{D}_u^b$.
      Calculate the basic critic target Q-value $z_{basic,u}^i$ according to (35).
      Update the basic critic network $\theta_{Q_u^b}$ by performing a gradient descent step with the gradient calculated by (36).
      Update the actor network θ_{μ_u} by performing a gradient descent step with the gradient calculated by (38).
    end for
    Update the target networks according to (39), (40) and (41).
  end for
end for

that the computation capacities of RDs, SDs and MEC servers range from [0.6, 1], [1.6, 2] and [6, 8] gigacycles/s, respectively [32]. For the computation task, the input data size is uniformly generated from [0.1, 0.3] Mbits and the required number of CPU cycles per bit is from 900 to 1100 cycles/bit [17]. We summarize our default simulation parameters in Table 2, if not specified.

TABLE 2
System Parameters

Parameter	Value
Number of mobile edge servers (M)	3
Number of RDs (U)	12
Number of SDs (V)	60
Cell coverage radius (R_{BS})	300 m
D2D range (R_{D2D})	50 m
Number of sub-channels (K)	10
Total bandwidth (B_{tot})	10 MHz
Noise power spectrum density	-174 dBm/Hz
Path loss exponent (ς)	4
Transmit power of RDs	0.2 W
Computation capacity of RDs ($f^t_{RD,u}$)	[0.6, 1] gigacycles/s
Computation capacity of SDs ($f^t_{SD,v}$)	[1.6, 2] gigacycles/s
Computation capacity of MEC servers ($f^t_{MEC,m}$)	[6, 8] gigacycles/s
Task input data size (b^t_u)	[0.1, 0.3] Mbits
Required number of CPU cycles per bit (c^t_u)	[900, 1100] cycles/bit

The actor, basic critic and meta critic networks in HDRL all have two fully connected feedforward hidden layers. The learning rates for the actor, basic critic and meta critic networks are 0.0001, 0.001 and 0.0001, respectively [29]. When the learning rate is too high, the algorithm may not converge and the loss will fluctuate. If the learning rate is too small, the convergence speed of the algorithm is slow, and thus longer training time is needed. The replay buffer sizes $\mathcal{D}^m$ and $\mathcal{D}^b_u$ are 10000, and the batch size is 200 [42]. For DNN, large batch size usually makes the network converge faster, but too large batch size may lead to insufficient memory or program kernel crash. We set the maximum time slot number $T = 20$ in each episode [29], the discount factor $\gamma = 0.99$, and the soft target update rate $\tau = 0.001$. The exploration noise follows a normal distribution with variance $\sigma_\upsilon = 0.1$ [41]. The exploration probability of ε-greedy policy is initialized as $\varepsilon = 0.5$ and decreases with the number of episodes [25].The setting of ε needs to balance the tradeoff between exploration and exploitation. The algorithm prefers exploration when ε is large, and the algorithm tends to exploit when ε is small.

The performance metrics we employ include the average task computation latency and energy consumption. The task computation latency and energy consumption of RD u are calculated by Eq. (16) and Eq. (17), respectively. To verify the effectiveness of HDRL, we compare it with the following four benchmark schemes:

1) *Local Execution (LE) scheme*: Each RD executes its computation tasks locally and computation offloading is not involved.

2) *Random Offloading (RO) scheme*: The offloading ratio, offloading node and sub-channel selection are decided randomly.

3) *Complete Offloading (CO) scheme*: Each RD offloads its computation tasks completely for D2D or MEC execution, while the offloading node and sub-channel selection are decided at random.

4) *DQN based (DQN) scheme*: As the action space of DQN can only be discrete values, we discretize the offloading ratio into five actions. Instead of the DDPG part of the proposed HDRL algorithm, DQN is used to output the discrete actions of offloading ratios for all discrete action pairs, and the meta critic network is still used to output the joint discrete action pairs.

First, the convergence property of the proposed HDRL algorithm is demonstrated in Fig. 3. From Fig. 3, we can observe that the reward increases as the number of episodes until it attains to a relatively stable value. It is shown that the proposed HDRL algorithm is convergent. Based on the convergence property of the proposed HDRL algorithm, the following simulation results are averaged over 5000 episodes for analysis.

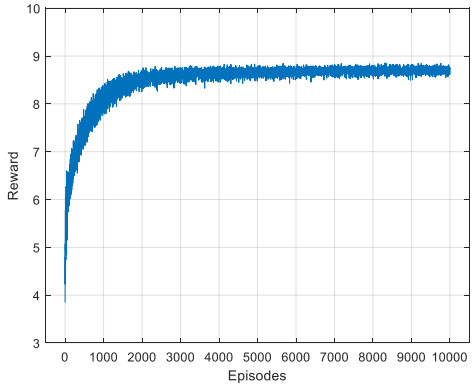
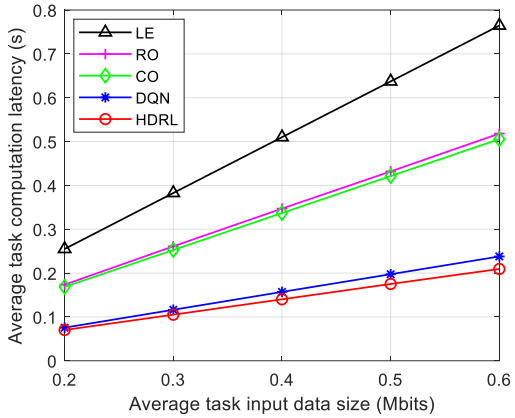


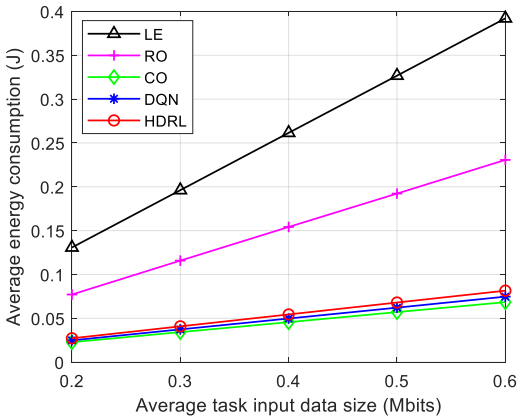
Fig. 3. Illustration for the convergence property of HDRL.

Next, we investigate the impact of the average task input data size on the average task computation latency and energy consumption in Fig. 4. It is shown that the average task computation latency and energy consumption increase with the task input data size for all the schemes, as larger task input data size leads to more latency and energy consumption for task execution and data transmission. In Fig. 4 (a), the HDRL and DQN schemes have lower increasing rate in task computation latency than other schemes, as the HDRL and DQN schemes learn to coordinate the decisions of offloading ratio, offloading node selection and sub-channel selection to achieve a better performance. In Fig. 4 (b), the average energy consumption of HDRL is lower than that of LE and RO. The reasons are as follows. In our simulation, the average energy consumption of local execution is higher than that of data transmission. Thus, LE has the highest energy consumption as it executes its tasks locally and data transmission is not involved. Moreover, the offloading ratio of RO is decided randomly while that of HDRL depends on the benefits of computation offloading. Since the benefits of computation offloading are significant, the average offloading ratio of HDRL is higher than that of RO. Thus, the average energy consumption of HDRL is lower than that of RO. Furthermore, with the intelligent decision of sub-channel selection, HDRL can significantly reduce the interference in data transmission, so as to reduce the energy consumption of data transmission. However, CO has the lowest energy consumption since it avoids the local execution cost. Note that HDRL outperforms the DQN scheme with 12% lower task computation latency at the cost of only 9% higher energy consumption. Since we aim at minimizing the total task computation latency, HDRL can make a better decision than DQN. In Fig. 4 (a), compared with the LE, RO, CO and DQN schemes, HDRL decreases the average task computation latency approximately by 73%, 60%, 59% and 12%, respectively, when the average task input data size is 0.6 Mbits.

Then, we examine the impact of the average required number of CPU cycles per bit on the average task computation latency and energy consumption in Fig. 5. As shown in Fig. 5, when the required number of CPU cycles per bit increases, the average task computation latency increases for all the schemes, and the average energy consumption increases for all the schemes except CO. This is due to the fact that with the increasing required number of CPU cycles per bit, the latency and energy consumption of task execution increase while that of data transmission remain unchanged. In Fig. 5 (a), it is shown that HDRL outperforms the LE, RO, CO and DQN schemes with the improvement on the average task computation latency approximately 73%, 57%, 49% and 5%, respectively, when the

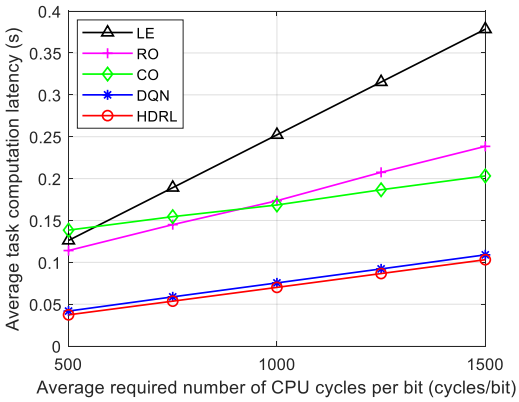


(a) Average task computation latency

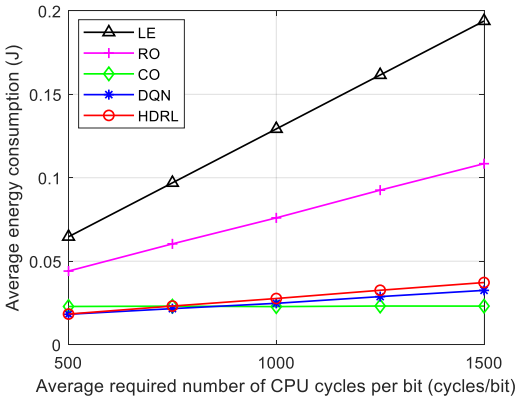


(b) Average energy consumption

Fig. 4. The average task computation latency and average energy consumption of the HDRL, LE, RO, CO and DQN schemes for different average task input data sizes.



(a) Average task computation latency



(b) Average energy consumption

Fig. 5. The average task computation latency and average energy consumption of the HDRL, LE, RO, CO and DQN schemes for different average required numbers of CPU cycles per bit.

average required number of CPU cycles per bit is up to 1500 cycles/bit.

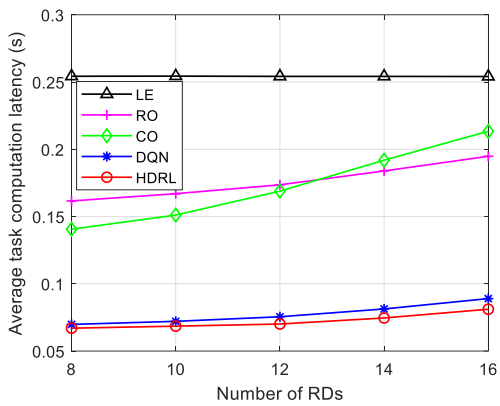
In the following, we compare the average task computation latency and energy consumption for different numbers of RDs in Fig. 6. As shown in Fig. 6, the average task computation latency and energy consumption increase with the number of RDs for all the schemes except LE, since the increase of RDs leads to increased resource contention. Since the HDRL scheme can intelligently coordinate the decisions of offloading ratio, offloading node selection and sub-channel selection, it achieves the lowest task computation latency than that of other schemes. In Fig. 6 (a), compared with the LE, RO, CO and DQN schemes, HDRL decreases the average task computation latency approximately by 68%, 58%, 62% and 9%, respectively, when the number of RDs is 16.

Fig. 7 shows the average task computation latency and energy consumption with respect to the average computation capacity of MEC servers. In Fig. 7, the performance of LE dose not change since it is independent of the computation capacity of MEC servers. The average task computation latency decreases with the increasing computation capacity of MEC servers for other schemes, as the MEC execution latency is improved. The average energy consumption decreases with the increasing computation capacity of MEC servers for the HDRL and DQN schemes, because more fraction of computation tasks can be executed at MEC servers to save the energy of local execution. In Fig. 7 (a), when the average computation capacity of MEC servers is up to 9 gigacycles/s, HDRL provides about 76%, 65%, 63% and 7% lower task computation latency than the LE, RO, CO and DQN schemes, respectively.

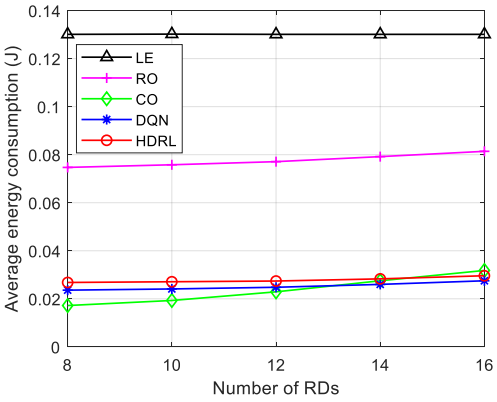
Finally, we compare the average task computation latency and energy consumption when the total bandwidth varies from 5 MHz to 25 MHz in Fig. 8. As shown in Fig. 8, the average task computation latency and energy consumption decrease with the total bandwidth for all the schemes except LE. The reasons are as follows. With the increasing total bandwidth, the latency and energy consumption for data transmission can be significantly reduced, while local execution does not require data transmission. In Fig. 8 (a), for total bandwidth of 25 MHz, HDRL can achieve 73%, 54%, 38%, and 6% less task computation latency than the LE, RO, CO and DQN schemes, respectively.

6 Conclusion

In this paper, we investigated the joint computation offloading and resource allocation problem for a D2D-assisted MEC partial offloading scenario, meanwhile taking into account the dynamics of both computation resources and channel conditions. We jointly formulated the partial computation offloading and resource allocation problem in shared spectrum with the aim of minimizing the total task computation latency in the long-term. We proposed an HDRL framework with a hierarchical actor-critic architecture to solve the problem, where actors use deep deterministic policy networks to output continuous actions, *i.e.*, offloading ratios, and a meta critic considers the outputs of all actors and learns to coordinate in offloading node and

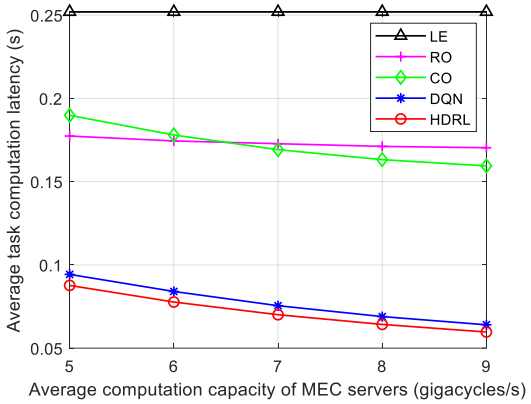


(a) Average task computation latency

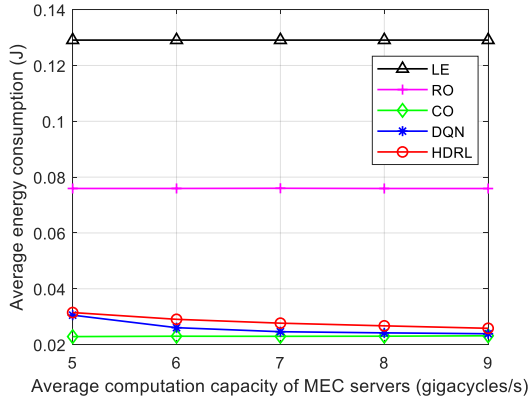


(b) Average energy consumption

Fig. 6. The average task computation latency and average energy consumption of the HDRL, LE, RO, CO and DQN schemes for different numbers of RDs.



(a) Average task computation latency



(b) Average energy consumption

Fig. 7. The average task computation latency and average energy consumption of the HDRL, LE, RO, CO and DQN schemes for different average computation capacities of MEC servers.

sub-channel selection. Simulation results demonstrated that our proposed HDRL based algorithm can achieve much better task computation latency performance than baseline offloading schemes.

For the future work, we will further develop some practical strategies for D2D-assistant MEC offloading systems. First, we have assumed that each RD is allocated with one orthogonal sub-channel. In the future work, efficient wireless bandwidth allocation scheme can be investigated to further improve the system performance for different types of applications, where joint computation offloading, wireless bandwidth allocation and interference management should be considered. Second, we intend to investigate the incentive mechanism to motivate the D2D cooperation among devices for computation offloading and communication resource sharing. Third, it is interesting to study the energy efficiency of D2D-assistant MEC offloading systems, *i.e.*, minimizing the energy consumption under the constraint on computation latency.

REFERENCES

[1] "Mobile application market size, share & trends analysis report by store type (google store, apple store), by application (gaming, music & entertainment, health & fitness), by region, and segment forecasts, 2020-2027," https://www.grandviewresearch.com/industry-analysis/mobile-application-market, Grand View Research, 2020.

[2] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[3] 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 of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[4] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for internet of everything," *China Communications*, vol. 16, no. 3, pp. 32–41, 2019.

[5] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

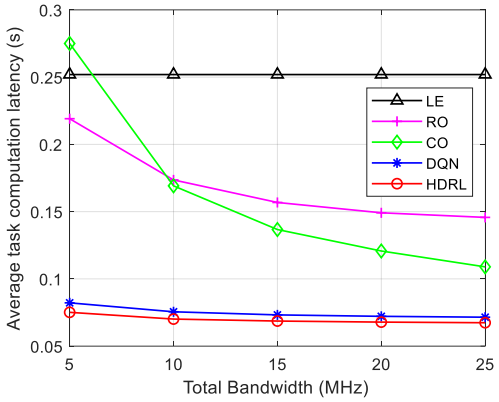
[6] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Hu, "iRAF: A deep reinforcement learning approach for collaborative mobile edge computing IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7011–7024, 2019.

[7] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

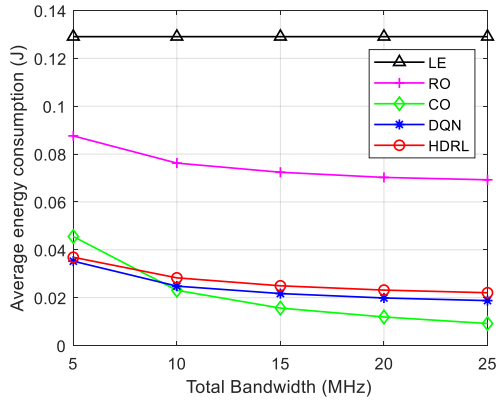
[8] Y. Hwang, S. Kang, and J. Shin, "A study of efficient small data transmission in industry IoT based 3GPP NB-IoT system," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 1848–1850.

[9] D. Chatzopoulos, C. Bermejo, E. u. Haq, Y. Li, and P. Hui, "D2D task offloading: A dataset-based Q&A," *IEEE Communications Magazine*, vol. 57, no. 2, pp. 102–107, 2019.

[10] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2D-assisted multi-



(a) Average task computation latency



(b) Average energy consumption

Fig. 8. The average task computation latency and average energy consumption of the HDRL, LE, RO, CO and DQN schemes for different total bandwidth.

user cooperative partial offloading, transmission scheduling and computation allocating for MEC," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 4858–4873, 2021.

[11] Y. Yang, C. Long, J. Wu, S. Peng, and B. Li, "D2D-enabled mobile-edge computation offloading for multiuser IoT network," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 490–12 504, 2021.

[12] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: A potential game approach," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3226–3237, 2022.

[13] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

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

[15] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[16] W. Zhang, R. Yadav, Y.-C. Tian, S. K. K. Sah Tyagi, I. A. Eelgendy, and O. Kaiwartya, "Two-phase industrial manufacturing service management for energy efficiency of data centers," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2022.

[17] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1750–1763, 2019.

[18] M. Sun, X. Xu, X. Tao, and P. Zhang, "Large-scale user-assisted multi-task online offloading for latency reduction in D2D-enabled heterogeneous networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2456–2467, 2020.

[19] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4193–4207, 2019.

[20] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472–4486, 2020.

[21] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 198–14 211, 2020.

[22] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018.

[23] R. Yadav, W. Zhang, I. A. Elgendy, G. Dong, M. Shafiq, A. A. Laghari, and S. Prakash, "Smart healthcare: RL-based task offloading scheme for edge-enable sensor networks," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 24 910–24 918, 2021.

[24] S. Huang, B. Lv, R. Wang, and K. Huang, "Scheduling for mobile edge computing with random user arrivals–An approximate MDP and reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7735–7750, 2020.

[25] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, 2020.

[26] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019.

[27] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[28] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 9, pp. 2745–2762, 2021.

[29] J. Zhang, J. Du, Y. Shen, and J. Wang, "Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9303–9317, 2020.

[30] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[31] W. Jiang, G. Feng, S. Qin, and Y. Liu, "Multi-agent reinforcement learning based cooperative content caching for mobile edge networks," *IEEE Access*, vol. 7, pp. 61 856–61 867, 2019.

[32] R. Chai, J. Lin, M. Chen, and Q. Chen, "Task execution cost minimization-based joint computation offloading and resource allocation for cellular D2D MEC systems," *IEEE Systems Journal*, vol. 13, no. 4, pp. 4110–4121, 2019.

[33] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile D2D networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1610–1622, 2019.

[34] M. Guo, W. Wang, X. Huang, Y. Chen, L. Zhang, and L. Chen, "Lyapunov-based partial computation offloading for multiple mobile devices enabled by harvested energy in MEC," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[35] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 137–12 151, 2018.

[36] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[37] G. Hu, Y. Jia, and Z. Chen, "Multi-user computation offloading with D2D for mobile edge computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[38] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5174–5184, 2018.

[39] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 158–11 168, 2019.
[40] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.
[41] R. Ding, Y. Xu, F. Gao, and X. Shen, "Trajectory design and access control for air-ground coordinated communications system with multi-agent deep reinforcement learning," *IEEE Internet of Things Journal*, pp. 1–1, 2021.
[42] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of things with decentralized cooperative edge caching," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9441–9455, 2020.

Wei Jiang received the B.S. degree in School of Communication and Information Engineering at Chongqing University of Posts and Telecommunications (CQUPT) in 2013, and the Ph.D. degree in School of Communication and Information Engineering at University of Electronic Science and Technology of China (UESTC) in 2019. She was a visiting Ph.D. student at the Pennsylvania State University (PSU) from 2017 to 2018. She is currently a postdoctoral researcher with the College of Electronics and Information Engineering, Shenzhen University. Her current research interests include next generation mobile communication systems, machine learning and content caching and distribution techniques in mobile networks.

Daquan Feng received the Ph.D. degree in information engineering from the National Key Laboratory of Science and Technology on Communications, University of Electronic Science and Technology of China, Chengdu, China, in 2015. From 2011 to 2014, he was a visiting student with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. After graduation, he was a Research Staff with State Radio Monitoring Center, Beijing, China, and then a Postdoctoral Research Fellow with the Singapore University of Technology and Design, Singapore. Since 2016, he has been with the College of Electronics and Information Engineering, Shenzhen University, as an Assistant Professor and then Associate Professor. His research interests include URLLC communications, MEC, and massive IoT networks. Dr. Feng is an Associate Editor of IEEE COMMUNICATIONS LETTERS.

Yao Sun is currently a Lecturer with James Watt School of Engineering, the University of Glasgow, Glasgow, UK. Dr. Sun has extensive research experience in wireless communication area. He has won the IEEE Communication Society of TAOS Best Paper Award in 2019 ICC. He has been the guest editor for special issues of several international journals. He has been served as TPC Chair for UCET 2021, and TPC member for number of international conferences, including VTC spring 2022, GLOBECOM 2020, WCNC 2019, ICCT 2019. His research interests include intelligent wireless networking, network slicing, blockchain system, internet of things and resource management in mobile networks. Dr. Sun is a senior member of IEEE.

Gang Feng (M'01, SM'06) received his BEng and MEng degrees in Electronic Engineering from the University of Electronic Science and Technology of China (UESTC), in 1986 and 1989, respectively, and the Ph.D. degrees in Information Engineering from The Chinese University of Hong Kong in 1998. He joined the School of Electric and Electronic Engineering, Nanyang Technological University in December 2000 as an assistant professor and was promoted as an associate professor in October 2005. At present he is a professor with the National Laboratory of Communications, University of Electronic Science and Technology of China.

Dr. Feng has extensive research experience and has published widely in computer networking and wireless networking research. His research interests include AI enabled wireless access and content distribution, next generation cellular networks, etc. Dr. Feng is a senior member of IEEE.

Zhenzhong Wang received the Ph.D. degree in electromagnetic field and microwave technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2010. Since 2010, he has been with the Technology and Management Center of China Central Television which is renamed as China Media Group in 2018. Now, Dr. Wang is a Professorate Senior Engineer. His research interests include 4K/8K UHD production, media contents distribution and 5G transmission.

Xiang-Gen Xia (M'97, S'00, F'09) received his B.S. degree in mathematics from Nanjing Normal University, Nanjing, China, and his M.S. degree in mathematics from Nankai University, Tianjin, China, and his Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1983, 1986, and 1992, respectively.

He was a Senior/Research Staff Member at Hughes Research Laboratories, Malibu, California, during 1995-1996. In September 1996, he joined the Department of Electrical and Computer Engineering, University of Delaware, Newark, Delaware, where he is the Charles Black Evans Professor. His current research interests include space-time coding, MIMO and OFDM systems, digital signal processing, and SAR and ISAR imaging. Dr. Xia is the author of the book Modulated Coding for Intersymbol Interference Channels (New York, Marcel Dekker, 2000).

Dr. Xia received the National Science Foundation (NSF) Faculty Early Career Development (CAREER) Program Award in 1997, the Office of Naval Research (ONR) Young Investigator Award in 1998, and the Outstanding Overseas Young Investigator Award from the National Nature Science Foundation of China in 2001. He received the 2019 Information Theory Outstanding Overseas Chinese Scientist Award, The Information Theory Society of Chinese Institute of Electronics. Dr. Xia has served as an Associate Editor for numerous international journals including IEEE Transactions on Signal Processing, IEEE Transactions on Wireless Communications, IEEE Transactions on Mobile Computing, and IEEE Transactions on Vehicular Technology. Dr. Xia is Technical Program Chair of the Signal Processing Symp., Globecom 2007 in Washington D.C. and the General Co-Chair of ICASSP 2005 in Philadelphia.