Wireless Networks
https://doi.org/10.1007/s11276-021-02643-w

ORIGINAL PAPER

Deep reinforcement learning-based computation offloading and resource allocation in security-aware mobile edge computing

H. C. Ke[1,2] · H. Wang[3] · H. W. Zhao[4] · W. J. Sun[3]

Accepted: 7 May 2021

Abstract

Owing to the insufficient processing ability of wireless devices (WDs), it is difficult for WDs to process these data within the deadline associated with the quality of service requirements. Offloading computation tasks (workloads) to emerging mobile edge computing servers with small or macro base stations is an effective and feasible solution. However, the offloaded data will be fully exposed and vulnerable to security threats. In this paper, we introduce a wireless communication and computation model of partial computation offloading and resource allocation considering the time-varying channel state, the bandwidth constraint, the stochastic arrival of workloads, and privacy preservation. To simultaneously optimize the computation and execution delays, the power consumption, and the bandwidth resources, we model the optimization problem as a Markov decision process (MDP) to minimize the weighted sum cost of the system. Owing to the difficult problems of lack of *priori* knowledge and the curse of dimensionality, we propose a decentralized optimization scheme on partial computation offloading and resource allocation based on deep reinforcement learning (DOCRRL). According to the time-varying channel state, the arrival rate of computation workloads, and the signal-to-interference-plus-noise ratio, the DOCRRL algorithm can learn the optimal policy for decision-making under stringent latency and risk constraints that prevent the curse of dimensionality from arising owing to the high-dimensional action space and state space. The numerical results reveal that DOCRRL can explore and learn the optimal decision-making policy without *priori* knowledge; it outperforms four baseline schemes in simulation environments.

Keywords Computation offloading · Quality of service · Mobile edge computing · Deep reinforcement learning · Privacy preservation

✉ H. Wang
email_wanghui@126.com

H. C. Ke
kehongchang1981@163.com

H. W. Zhao
27376097@qq.com

W. J. Sun
sunweijia_2000@126.com

[1] School of Computer Technology and Engineering, Changchun Institute of Technology, Changchun, China

[2] National and Local Joint Engineering Research Center for Intelligent Distribution Network Measurement, Control and Safe Operation Technology of National Development and Reform Commission, Changchun Institute of Technology, Changchun, China

[3] College of Computer Science and Engineering, Changchun University of Technology, Changchun, China

[4] College of Computer Science and Technology, Jilin University, Changchun, China

Published online: 17 May 2021

1 Introduction

Although the battery capacity and computing power of wireless devices (WDs) have improved greatly because of advances in processor technology and manufacturing standards, their processing ability is still insufficient to satisfy the quality of service (QoS) considering numerous computationally intensive or delay-sensitive computation workloads and large amount of computing data. Because of exponential growth in the amount of computing data generated by WDs under stringent deadlines, the battery capacity and resource constraints remain a bottleneck [1, 2]. Understanding how to meet current computing needs so as to address this problem is a key challenge [3]. In previous years, cloud computing, which offers tremendous computing resources, was able to meet the computing demand of WDs and improve user experience. There has been some research on computation task offloading, wireless resource allocation in cloud computing. Hadded et al. proposed a method of computation offloading of computationally intensive workloads to the cloud server in cloud computing environments combining the fog computing [4]. And Du et al. designed a mobile cloud computing model to relieve the energy consumption and increase the computational efficiency of WDs [5]. However, cloud computing servers cover very large regions and are often far from the WDs with workloads that need to be tackled. Consequently, reducing the transmission time and backhaul time of computing data is an intractable problem.

The emerging technology of mobile edge computing (MEC) can provide a promising approach by combining cloud computing and fog computing [6]. MEC servers are typically geographically closer to WDs and have much more computing power than WDs [7]. Thus, MEC can solve not only the problem of the transmission delay in cloud computing but also the problem of WD resource constraints [8]. The workloads from WDs can be offloaded to MEC server connected to macro base station (MBS) on the deadline constraint. However, the channel state between the MEC and WDs should be considered to satisfy the wireless transmission requirements. Thus, bandwidth allocation and offloading decisions for WDs in a time-varying channel state are urgent problems.

By exploiting the advantages of MEC, WDs can offload computation workloads to MEC servers while accommodating channel conditions and latency constraints [9, 10]. Binary offloading is typically applied to meet the computational demand; that is, WDs covered by MEC can process computation workloads by local execution (no offloading) or offloading to an MEC server (full offloading). However, binary offloading is not feasible in some communication and computation environments when the workloads are too large for computation offloading or the bandwidth is insufficient to meet the communication requirements of all the WDs [11]. In heterogeneous networks with many WDs, partial execution of computation workloads by WDs and partial offloading to the MEC server is more reasonable. In addition, offloading workloads to edge server will inevitably involve exposure to security threats such as denial-of-service attacks, snooping, and alteration. Therefore, some security service algorithm can be used to avoid these threats, thereby ensuring the security of computation offloading. Security algorithms such as symmetric encryption or homomorphic encryption [12, 13] have been applied for data protection. However, these schemes do not consider the constraints on computing resources during offloading to the server. Moreover, the security service algorithm will use additional resources of the node and the MEC server, resulting in delays and increased energy consumption. Therefore, it is essential to design a reasonable and effective task security service policy and meet the task offload requirements.

In an MEC model with multiple WDs, the channel state between the MEC server and WDs is usually time-varying, and the bandwidth is not unlimited. We consider a changing channel with multiple WDs and investigate bandwidth allocation to each WD to meet the transmission and execution deadlines and the required security level. Moreover, depending on the time-varying channel state and the arrival rate of the computation workloads, it is critical to select the execution mode and bandwidth allocation ratio that provide the optimal scheme for obtaining the minimum weighted sum cost. The binary offloading mode has been used in many works to address this optimization problem by mixed integer programming (MIP). However, MIP cannot be implemented using the conventional convex optimization method but needs to be broken down into multiple sub-problems. Other schemes are available for solving offloading optimization problems; for example, a branch and bound algorithm has been proposed [14], and a dynamic programming scheme has been designed [15] to locally execute or offload workloads to an MEC server. Moreover, the heuristic greedy algorithm has been used to make the offloading decision so as to decrease the computational complexity caused by the large number of WDs. However, the aforementioned algorithms are not applicable to the unstable channel condition, and when the number of WDs is too large, they are difficult to converge owing to excessive iteration.

In light of recent research on deep reinforcement learning (DRL), and considering the stochastic size of workloads and the time-varying channel condition, a decentralized optimization scheme for partial computation offloading and resource allocation based on deep reinforcement learning (DOCRRL) are proposed. Each WD

has a learning agent to interact with the proposed MEC environment, then observe the state independently. This scheme simultaneously optimizes the delay, energy consumption, bandwidth, and privacy and security cost and yields the minimum weighted sum cost. Compared with the previous works, we summarize our following contributions:

- We model a novel MEC framework connected to multiple WDs in cellular networks considering the need for secure offloading. In the implemented model, the channel state between MBS and WDs is time-varying, and the workloads generated by WDs arrive stochastically. We adopt round-robin MEC server activation after a certain time interval. The partial offloading ratio, as well as bandwidth allocation ratio are selected to minimize the total cost, including the computation and execution delays, bandwidth, energy consumption, and privacy and security cost.
- We design the optimization problem for partial offloading and resource allocation as a Markov decision process (MDP) and define the state space, action space, and reward function. When interacting with the environment, the agent of each WD fully takes into account the changing channel state and stochastic arrival of workloads.
- To tackle the curse of dimensionality caused by massive WDs, we propose the DRL-based algorithm (DOCRRL) to solve the offloading decision-making and resource allocation optimization problem. DOCRRL leverage MEC severs with the greatest remaining computing capacity or inactive mode to train the neural network (NN) for each WD, and can receive the minimum weighted sum cost under stringent latency and security constraints in stochastic environments.
- We perform extensive numerical simulations of the proposed MEC system taking into the time-varying channel state and available bandwidth account to reveal effectiveness, and performance of DOCRRL compared with four benchmark algorithms. The decentralized scheme of DOCRRL can dramatically improve the convergence performance since it has nothing to do with the number of WDs that generate tasks.

The remainder of this paper is organized as follows. Section 2 reviews related works. A security-aware MEC framework is presented in Sect. 3. Section 4 describes in detail the DRL-based optimization scheme. Extensive numerical results are presented in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 Related works

Currently, there are some existing works on computation offloading and resource allocation in wireless networks or vehicular networks. In this section, we introduced related works from three aspects: Computation offloading and wireless resource allocation, optimization scheme based on RL or DRL for computation offloading or resource allocation, and privacy preservation and security in wireless networks.

2.1 Computation offloading and wireless resource allocation

There had been considerable research on computation offloading. Valentino et al. designed an opportunistic computation offloading model based on a neural network (NN) by borrowing computing resources from other devices. This offloading method made full use of the computing resources of device clusters, which improved the computational efficiency of the devices [16]. Lyu et al. implemented an offloading strategy to obtain the minimum energy consumption of all terminal. The designed scheme adaptively selected offloading opportunities by self-nomination or self-denial [17]. However, these methods were based on binary offloading, which is not feasible when there is a large number of computation workloads. However, several works have considered partial offloading. For instance, Ning et al. introduced a partial computation offloading method with no MEC resource constraint considering the combination of cloud computing and edge computing [18]. Tang et al. formulated the partially observable offloading decision-making issue as a decentralized partially observable Markov decision process for maximizing the total utility under the latency constraint [19]. However, these works on partial offloading did not consider the resource constraints of the MEC or the time-varying channel state. Regarding resource allocation, Wang et al. designed an optimization problem for offloading decisions and the data caching policy; they then decomposed optimization problem into subproblems in a distributed and efficient way [20]. Qian et al. investigated simultaneous optimization of the order of successive interference cancellation and computation resource allocation to Internet of Things(IoT) devices in a narrowband IoT system [21]. Tan et al. formulated a virtual resource allocation scheme in which user association, power control, caching and computation offloading policies, and resource allocation were simultaneously considered [22]. Wu et al. proposed an online resource allocation strategy for an MEC network that considered the time-varying demands of the combinatorial resources of mobile users [23]. Some of

these works considered the time-varying remained resources or the bandwidth of MBS, but they did not take into account the time-varying channel state.

2.2 Optimization scheme based on RL or DRL

With the rapid development of DL, applications of RL or DRL to computation offloading and resource allocation had been proposed. Wang et al. formulated an offloading selection scheme for an MEC environment as an MDP and used the directed acyclic graph to design an offloading policy[24]. Ning et al. implemented an energy-efficient computational offloading problem based on Deep Q-network (DQN) that minimized the overall power consumption while satisfying the delay constraint [25]. Liu et al. introduced a vehicle-assisted offloading method for user equipment that considered the computation delay of arriving workloads; they then formulated an optimization solution to require the maximum utility based on traditional Q-learning and DRL [26]. Huang et al. proposed an MEC model with a wireless power supply and then formulated a binary offloading and bandwidth allocation scheme. For the offloading policy, this work used DRL to learn the optimal selection scheme; the optimal bandwidth allocation was based on the optimal offloading scheme [27]. Chen et al. modelled the offloading decision-making processing as an MDP to implement the maximization of the long-term utility using Double DQN. [28] technique. Wang et al. considered the tradeoff between transmission, processing latency and the energy consumption of WDs in vehicular networks and addressed cost minimization using an MDP; then a binary offloading optimization scheme was solved by DRL [29]. Most existing works based on DRL have focused on only the particular type of computation workloads and assumed that task offloading was not affected by the task size or that the communication network environment was known by the DRL agent. Moreover, when the number of WDs increased, existing DRL schemes did not effectively avoid the curse of dimensionality.

2.3 Privacy preservation and security

Xiao et al. introduced a malware detection policy based on Dyna-Q algorithm under the cloud computing and edge computing environment. The proposed algorithm can enhance the accuracy of malware detection [30]. He et al. presented a constrained MDP-based privacy-aware offloading method for a healthcare IoT device with energy harvesting. The scheme adjusted the offload policy in light of the location privacy and user privacy, and improved the privacy level [31]. Min et al. improved the scheme proposed by He et al. [31] and devised an RL-based post-decision state learning policy to increase the offloading ratio. A model of location privacy and user privacy was also introduced to reduce the offloading rate under good channel conditions and increase the offloading rate under poor channel conditions to protect user privacy [32]. Elgendy et al. proposed an integer-linear-optimization-based computation offloading scheme using an advanced encryption standard (AES) cryptographic technique to obtain the minimum cost of energy consumption and the latency [33].

In summary, there were few works on security-aware and privacy-preserving schemes for MEC or fog computing, especially for multiple mobile users. In addition, most works used only a single encryption algorithm to protect data privacy, which was not suitable for the wireless communication and computation scene when the channel states are time-varying or the arrival workloads are stochastic; thus, the reliability of these solutions was not convincing and reliable.

By contrast, we considered the time-varying channel state and stochastic task arrival. For the offloaded workloads, we used a round-robin encryption scheme with different encryption algorithms and then applied the proposed DOCRRL algorithm. When the computation workloads arrive, DOCRRL selected to partially offload workloads to MEC and partially execute locally at WDs. In addition, the bandwidth could be adaptively allocated for computation offloading.

3 System model

This section firstly introduces the network model for MEC communication and computation framework, then wireless communication model and bandwidth resource allocation model are implemented. Finally, we describe the computation model in detail from three aspects: local execution, computation offloading, and privacy-preserving model.

3.1 Network model

A communication and computation system consisting of multiple WDs, an MEC server cluster, and a security server are designed. As shown in Fig. 1, in wireless networks, WDs with sensors that aid data collection such as smart phones, IoT nodes, cameras, or watches, and are covered by the MEC server cluster. Each MEC server links to an MBS by an optical fiber link for receiving and transmitting computation workloads. At the same time, the computational ability of some WDs are limited and may not be sufficient for task computation. MEC servers with high-performance processors are located near the WDs; therefore, MEC can be fully exploited to perform computation workloads from the WDs as long as the WDs are within the

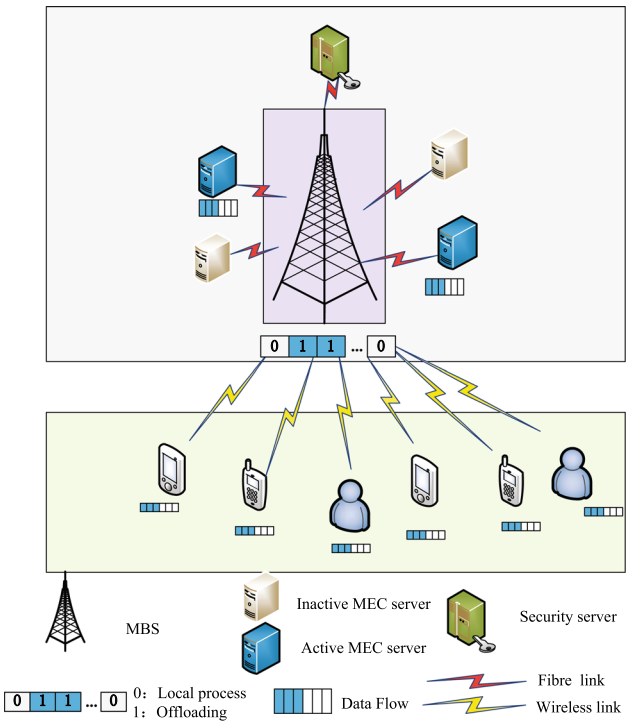


Fig. 1 MEC system model with multiple WDs

coverage of the certain MEC. However, if a large number of workloads generated by the WDs are all offloaded to the MEC servers, the resources or bandwidth may be insufficient, and congestion will occur. In the designed model, the stochastic and computationally intensive workloads continuously generated by the WDs can be either executed locally or offloaded to the MEC servers with high-performance processors by the MBS. To ensure secure data transmission, a security server is used to coordinate the encryption and decryption during data transmission. Without loss of generality, we denote $\mathcal{W} = \{1, 2, \cdots, W\}$ as the set of WDs, $\check{s}$ stands for the security server, and $\mathcal{M} = \{1, 2, \cdots, M\}$ is the set of MEC server clusters. Furthermore, there are W WDs in the MEC system and M MEC servers, an MBS. However, some MEC servers are not activated, and the WDs can communicate(offload workloads) only with active MEC servers. Thus, each MEC server m has a task buffer queue that stores computing data received from the WDs. In this MEC model, we define L_m as the size of the buffer queue and assume that WD w has a computation workload for processing and is covered by MBS of MEC server m, and that the computation workload can be split into n sub-workloads. Here, $\mathcal{N} = \{1, 2, \cdots, N\}$ is defined as the sub-workloads set, where $n \in \mathcal{N}$, and $w \in \mathcal{W}$. sub-workload n can be locally executed at WD w or offloaded to MEC server m by MBS. Table 1 shows the main notations and definitions in our work.

Table 1 notations in this paper

Notation	Definition
$\mathcal{W}$	Set of WDs
$\mathcal{M}$	Set of MEC servers
$\check{s}$	Security server
$\mathcal{T}$	Index of total time slots
$H_{m,w}(t)$	Channel vector between WD w and MBS m
$h_{m,w}(t)$	Total channel gain
ρ_c	Normalized correlation coefficient
$e_c(t)$	The error vector
B_m	The communication bandwidth
$P_{l,w}(t)$	Allocated power when local processing
$P_{m,w}(t)$	Transmission power when offload to the MBS
$ts_w(t)$	Size of arriving workloads from WD w
ζ_w	SINR of the MBS
τ_w	Allocated bandwidth for WD w
λ_w	Arrival rate of workloads from WD w
$\mathcal{S}$	State space of the proposed DOCRRL
$\mathcal{A}$	Action space of the proposed DOCRRL
π	The policy
$Q(\mathcal{S}, \mathcal{A})$	State-action function

3.2 Communication model

As mentioned in Sect. 3A, the channel state between WD w and MBS m is varied in a continuous decision epoch. Here, we divide the sequential process time into discrete time slots (or epochs), and each time slot t is a constant. We denote the index set of time slots as $\mathcal{T} = \{1, 2, \cdots, T\}$. Each discrete time slot is set to 1, which is 1 delay unit. Here, 1 delay unit equals 10 ms because an MEC server with a high-performance processor has such a high computing speed that we can ignore the backhaul time for task computation offloading. As mentioned in Sect. 3A, we assume that the computation workloads to be executed are computationally intensive, as well as the uplink communication mode is multiple-input multiple-output. The channel vector $H_{m,w}(t)$ is denoted by

$$H_{m,w}(t) = \left[h_{m,w}^{(1)}, h_{m,w}^{(2)}, \cdot, h_{m,w}^{(\widetilde{w})}\right] \tag{1}$$

where $H_{m,w}(t)$ exhibits Rayleigh fading, and $\widetilde{w}$ is the total number of working antennas. However, we do not use a Markov chain to simulate the transformation process of the channel gain; rather, the time correlation autoregressive [34, 35] are utilized in this model. Thus, the transformation that is processed between two adjacent time intervals is defined by

$$H_{m,w}(t) = \rho_c H_{m,w}(t - t') + \sqrt{1 - \rho_c^2} e_c(t) \quad (2)$$

where ρ_c is the normalized correlation coefficient between $t - t'$ and t; ρ_c is close to 1. In addition, $e_c(t)$ is the error vector; $e_c(t) \sim CN(0, I_m)$, which has a complex Gaussian distribution and is uncorrelated with $H_{m,w}(t)$.

We have explained the interference among WDs in coverage region of MEC m in detail in our previous paper, so it is not described here [36]. The signal-to-interference-plus-noise ratio (SINR) of WD w in time slot t can be written as

$$\zeta_w(t) = \frac{P_{m,w}(t) \cdot \left| a_w^H(t) h_{m,w}(t) \right|^2}{\sum_{j \neq d, j \in \mathcal{D}} \left[P_{m,j}(t) \cdot \left| a_w^H(t) h_{m,j}(t) \right|^2 + \| a_w(t) \|^2 \right]} \quad (3)$$

where a_w and $h_{m,w}$ are the w-th columns of the matrices A and H, respectively. H is the $M \times W$ channel matrix, and A is the zero-forcing linear detector matrix associated with H [34, 35].

3.3 Resource allocation model

As mentioned in Sect. 3A, although the MEC servers are connected by an optical fiber link to the MBS, the bandwidth of MBS is not unlimited. That is to say, when the number of connected WDs increases, there will be some communication pressure on the bandwidth. We consider dividing the bandwidth into subbandwidths. The total bandwidth of the MBS m is B_m. We assume that $\tau_w(t)$ denote as the allocated ratio of bandwidth for WD w. The achievable transmitting rate between WD w and the MBS m is described by

$$r_{m,w}(t) = \tau_w(t) B_m \log_2(1 + \zeta_w(t)) \quad (4)$$

3.4 Computation model

As mentioned in Sect. 3A, the computation workloads to be processed can be split into subworkloads, which are computationally intensive and time-varying[1]. We assume that the arrival rate of workloads in each time slot follows Poisson distribution with λ_w. Without loss of generality, we define the computation workload as $A_w(t) \doteq (ts_w(t), c_w(t), T_{w,max}(t))$ in the time slot t, where $ts_w(t)$ stands for the size (in megabits-Mb) of workloads for WD w. Similarly, $c_w(t)$ represents the required number of CPU cycles to complete workloads for WD w. $T_{w,max}(t)$ gives the deadline for completing the workloads. λ_w is independently identically distributed in each time slot, and thus $\mathbb{E}[ts_w(t)] = \lambda_w$. Regarding the activation status of the MEC servers in $\mathcal{M}$, we define the *epoch* as the interval of an activated MEC server. During time slot $\mathcal{T}$, there are E epochs, and $\mathcal{E} = \{1, 2, \cdots, E\}$. The total duration of each epoch is $\lfloor T/E \rfloor$. We set T_e as the total time slot of the e-th epoch, and during T_e, the same MEC servers remain active. That is, the activation status of the MEC servers varies only between epochs.

3.4.1 Local execution

When the computational power and battery capacity of a WD are sufficient for execution, the computation task can be allocated for local processing. However, in this work, we consider that each computation workload is composed by some sub-workloads either completed by the WD or transmitted by MBS to certain edge server. Without loss of generality, we denote the offloading ratio for a computation task of WD w as $\eta_w(t)$. For WD w in time slot t, the allocated number of CPU for local processing is computed by

$$f_w(t) = \sqrt{\frac{P_{l,w}(t)}{\kappa \cdot (1 - \eta_w(t)) \cdot c_w(t)}} \quad (5)$$

where $P_{l,w}(t)$ denotes allocated power for local execution. And κ is the effective switched capacitance, which depends on the chip architecture of WD w.

Furthermore, we can derive the execution time required for local processing as

$$d_{l,w}(t) = \frac{(1 - \eta_w(t)) \cdot c_w(t)}{f_w(t)} \quad (6)$$

Then the energy consumption of WD w is described as

$$E_{l,w}(t) = P_{l,w}(t) \cdot d_{l,w}(t) \quad (7)$$

3.4.2 Computation offloading

If the computing data(workload) are selected to offloading execution, the activated MEC servers are allocated to execute the offloaded workload. In addition, the sub-workloads of WDs are independent, and there is no interference. The transmission time of computation workloads for WD w that are offloaded to the active MEC server m is obtained as

$$d_{m,w}^{tran}(t) = \frac{\eta_w(t) \cdot ts_w(t)}{r_{m,w}(t)} \quad (8)$$

Then the energy consumption required to transmit the computing data of WD w to m is defined as

$$E_{m,w}^{tran}(t) = P_{m,w}(t) \cdot d_{m,w}^{tran}(t) \quad (9)$$

[1] We assume that the computation workloads are a batch images to be processed, such as 1 batch of 10 traffic scene images or face recognition images.

where $P_{m,w}(t)$ is the transmission power for offloading workloads from WD w.

Next, we consider the execution time required to complete the received workloads by MEC server m. We denote the time factor for task processing at the MEC server as $\xi_1 = \{\xi_{1,1}, ..., \xi_{1,M}\}$, which is an M-dimensional vector for each MEC server and is proportional to $ts_w(t)$. The execution time for the computation task of WD w in the MEC server is

$$d_{m,w}^{exec}(t) = \xi_{1,m} \cdot \eta_w(t) \cdot ts_w(t) \tag{10}$$

Then the energy consumption for computation workloads of WD w in the MEC server is described as

$$E_{m,w}^{exec}(t) = P_{m,w}(t) \cdot d_{m,w}^{exec}(t) \tag{11}$$

The total time required for computation offloading from WD w can be obtained as

$$d_{m,w}(t) = d_{m,w}^{tran} + d_{m,w}^{exec}(t) \tag{12}$$

The total energy consumption of computation offloading for WD w can be obtained as

$$E_{m,w}(t) = E_{m,w}^{tran}(t) + E_{m,w}^{exec}(t) + E_{m,w}^{wait}(t) \tag{13}$$

where $E_{m,w}^{wait}(t)$ is the cost of waiting for execution by the MEC server. $\xi_2 = \{\xi_{2,1}, ..., \xi_{2,M}\}$ is a constant coefficient. According to Little's law, $E_{m,w}^{wait}(t) = \xi_{2,m} \cdot L_m(t)$.

3.5 Privacy-preserving model

During offloading, data from the WDs will be fully exposed to many security threats. Therefore, it is essential that the WDs and edge server implement mutual authentication or encryption to protect the privacy of the WD users. There are many encryption methods, such as symmetric encryption, asymmetric encryption, and homomorphic encryption. As shown in Fig. 2, taking into account the characteristics of the encryption algorithms, we use a round-robin encryption method for data encryption. Like the activation status of the MEC servers, the chosen encryption algorithm does not change during an epoch, and the encryption algorithm varies only between epochs.

We define $\sigma_{k,w}$ as the encryption policy for the computation workloads of WD w, and $\sigma_{k,w} \in \{0, 1\}, k \in \{1, 2, 3\}$. We define the delay cost of encryption as being proportional to $ts_w(t)$. Then, the delay cost of encryption is defined as

$$d_w^{en}(t) = \frac{\sigma_{k,w}\varsigma_k(1 - \eta_w(t)) \cdot c_w(t)}{f_w(t)} \tag{14}$$

where ς_k is the delay coefficient of the algorithm used for data encryption.

Thus, the energy consumption cost is given by

$$E_w^{en}(t) = P_{l,w}(t) \cdot d_w^{en}(t) \tag{15}$$

Similarly, the delay cost of decryption is derived as

$$d_w^{de}(t) = \sigma_{k,w}\varsigma_k'\eta_w(t)ts_w(t) \tag{16}$$

The energy consumption cost of decryption can be written as

$$E_w^{de}(t) = P_{m,w}(t) \cdot d_w^{en}(t) \tag{17}$$

To evaluate the privacy level, we assume that the risk rate is a function of the security level for different computation workloads. However, the security level of the security server, $\upsilon_{k,v}'$, does not change when the encryption algorithm is fixed. The task security level of the WD, $\upsilon_{k,v}$, varies during a time slot. We assume that the task security level follows the uniform distribution. We set $P(\upsilon_{k,w})$ as the risk rate of WD w in encryption mode k. Then $P(\upsilon_{k,w})$ can be written as

$$P(\upsilon_{k,w}) = \begin{cases} 0, & \upsilon_{k,w} \le \upsilon_{k,w}' \\ 1 - e^{-\lambda_k'(\upsilon_{k,w} - \upsilon_{k,w}')}, & \upsilon_{k,w} > \upsilon_{k,w}' \end{cases} \tag{18}$$

where λ_k' is the coefficient of the algorithm for controlling the risk rate.

The computation workloads offloaded by WDs have different security coefficients λ_k within a reasonable interval. It can be seen that when the task security level is less than that of the security server, the risk rate is 0, and thus the security server can execute the offloaded workloads to ensure the security level.

To meet the required offloading efficiency, not all offloaded workloads are risk-free during transmission and execution. Therefore, the risk rate $P(\upsilon_{k,w})$ meets the constraint of the risk probability, $P_{k,max}$, namely, $P(\upsilon_{k,w}) \le P_{k,max}$.

4 Problem statement and DRL-based scheme

Here, we describe in detail the problem statement and the design of the optimal scheme for computation offloading in the proposed MEC framework.

4.1 Problem statement

In Sect. 3, we firstly defined the latency and energy consumption of WD w. However, the goal of our work is to explore the optimal offloading decision-making and bandwidth allocation strategy for the minimization of the weighted sum cost, including the latency, energy

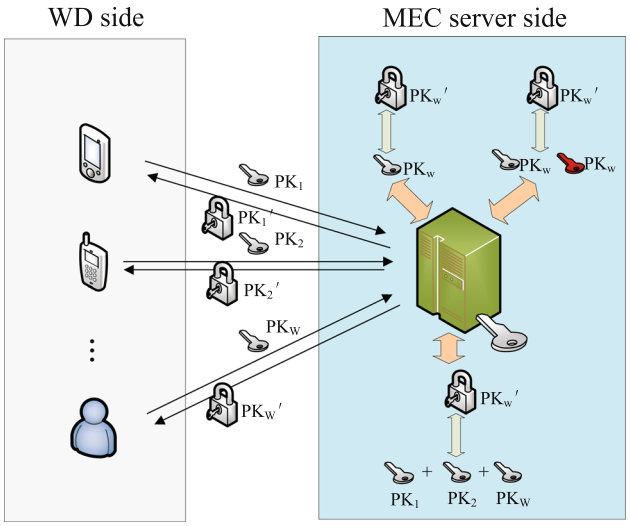


Fig. 2 Privacy-preserving model

consumption, bandwidth cost, and privacy and security cost for all WDs and MEC servers. Furthermore, as described in Sect. 3, the weighted delay cost C_d of local execution and offloaded execution for the arrival workloads is defined as

$$C_d(t) = \sum_{w=1}^{W} \left[\alpha_1 d_{l,w}(t) + a_2 \max_{m\in\hat{M}}\{d_{m,w}(t)\} + \alpha_3\left(d_w^{en}(t) + d_w^{de}(t)\right)\right] + P_w \tag{19}$$

where $\alpha_1, \alpha_2, \alpha_3$ are the coefficients that control the cost functions of local execution, offloaded execution, and privacy and security, respectively. $\hat{M}$ stands for the number of active servers, P_w is the penalty for the failure of task execution in time slot t, which satisfies $P_w = \mathbf{P}_{\{d_{l,w} > T_{w,max} \vee \{d_{m,w}^{off} > T_{w,max}\}}$, $d_{m,w}^{off}$ is the sum of the offloading and security delays.

The total energy cost C_m of local execution and offloaded execution by the MEC server for all the workloads is

$$C_m(t) = \sum_{w=1}^{W}\left[\alpha_1 E_{l,w}(t) + \sum_{m=1}^{\hat{M}} a_2 E_{m,w}(t) + \alpha_3\left(E_w^{en}(t) + E_w^{de}(t)\right)\right] \tag{20}$$

As far as the latency and energy costs are concerned, we need to consider the cost of the bandwidth and MBS. We assume that the bandwidth cost is proportional to the allocated bandwidth for offloading execution. Therefore, the bandwidth cost is defined by

$$C_b(t) = \sum_{w=1}^{W} \alpha_4 \tau_w(t) B_m + \alpha_5 \mathbf{I}_{\{w\in mbs\}} \tag{21}$$

where α_4 is the coefficient associated with the bandwidth cost, and α_5 represents the coefficient depends on the cost of the MBS. $\mathbf{I}$ is a cost function; its value is 1 when WD w is connected to the MBS.

Thus, the weighted sum cost for the proposed framework can be written by

$$C(t) = \omega_1 \cdot C_d(t) + \omega_2 \cdot [C_m(t) + C_b(t)] \tag{22}$$

where ω_1 and ω_2 are penalty weights associated with the delay cost and the costs of energy consumption, bandwidth, and the base station; it satisfies $\omega_1 + \omega_2 = 1$.

We need to minimize $C(t)$ under the constraints, which is defined as follows.

$$\mathbf{P1} \quad \min_{(\eta_w, \tau_w)} \quad \lim_{t\to\infty} \frac{1}{T}\sum_{t=1}^{T} C(t) \tag{23a}$$

$$\text{s.t.} \quad d_{l,w}(t) \le T_{w,max}(t), d_{m,w}(t) \le T_{w,max}(t) \tag{23b}$$

$$0 \le \tau_w(t) \le 1, \sum\nolimits_{w=1}^{W} \tau_w(t) = 1 \tag{23c}$$

$$0 \le \eta_w(t) \le 1 \tag{23d}$$

$$0 \le f_w(t) \le f_{max} \tag{23e}$$

$$P(\upsilon_{w,k}) \le P_{k,max} \tag{23f}$$

where (23b) indicates that the delay of local execution and offloaded processing must not exceed the maximum delay constraint. Equation (23c) is the ratio of the bandwidth allocated to WD w, which is less than 1. Equation (23d) is the offloading ratio of the workloads from WD w, which is less than 1. Equation (23e) shows that the allocated CPU cycles constraint on local processing does not exceed the maximum CPU cycles of the local processor. Equation (23f) ensures that the risk rate of offloading meets the constraint of the maximum risk probability.

We can see that **P1** is difficult to tackle by traditional convex optimization methods. Therefore, we consider using DRL to find the optimal decision-making strategy for **P1**.

4.2 Design of the optimal policy

However, the decision-making scheme for optimal computation offloading and bandwidth allocation to obtain the minimum weighted sum cost of proposed framework can be designed as an MDP. The agent observe the current state in the state space by interacting with the proposed MEC framework, then select an action based on the certain policy, further obtain an immediate reward from environment, and move to the next state [37, 38]. This process is repeated in the time slot space $\mathcal{T}$.

First, we define the state space, action space, and reward function. The state includes the channel vector, SINR, and security level of the arriving computation task. We define

the state space as $\mathcal{S}$. The state $s_(t) \in \mathcal{S}$ of each WD can be described as

$$\begin{aligned} s_t = \{ & \\ & \zeta_1(t), \zeta_2(t), \ldots, \zeta_w(t), \ldots, \zeta_W(t), \\ & H_{m,1}(t), H_{m,2}(t), \ldots, H_{m,w}(t), \ldots, H_{m,W}(t), \\ & A_1(t), A_2(t), \ldots, A_w(t) \ldots, A_W(t), \\ & \upsilon_{1,1}(t), \upsilon_{2,1}(t), \ldots, \upsilon_{k,1}(t) \ldots, \upsilon_{1,2}(t), \ldots, \upsilon_{K,W}(t) \\ \} & \end{aligned} \tag{24}$$

The action is composed of two dimension vectors: the offloaded ratio of the computation workloads, and the allocated ratio of the bandwidth, which is denoted as $\mathcal{A}$. The action $a_(t) \in \mathcal{A}$ in time slot t is written by

$$\begin{aligned} a_t = \{ & \\ & \tau_1(t), \tau_2(t), \ldots, \tau_w(t), \ldots, \tau_W(t), \\ & \eta_1(t), \eta_2(t), \ldots, \eta_w(t), \ldots, \eta_W(t), \\ \} & \end{aligned} \tag{25}$$

According to (19), the optimization scheme is to minimize the weighted sum cost in proposed MEC system. We can define the object function as a reward function that is the negative mean of the total cost. The reward function in time slot t is defined as $r_t = -C(t)/W$.

Here, the optimal scheme can be defined as an MDP, and we assume that the transition probability from s_t during two adjacent time slots is $P\{s'_t \mid s_t, \pi\}$, which is calculated by

$$\begin{aligned} & P\{s'_t \mid s_t, \pi\} \\ & \quad = P\{H'_{m,w} \mid H_{m,w}, \pi\} \times P\{\zeta' \mid \zeta_w, \pi\} \times \\ & \qquad P\{A'_{m,w} \mid A_{m,w}, \pi\} \times P\{\upsilon'_{k,w} \mid \upsilon_{k,w}, \pi\} \end{aligned} \tag{26}$$

where π is the policy of agent for WDs which is the notation of $\pi(a_t|s_t)$ and a mapping from $\mathcal{S}$ to $\mathcal{A}$.

Therefore, based on the policy π, the total cumulative reward can be written as

$$Q(s_t, a_t) = R_t + \gamma \cdot \sum_{s'_t \in S, a' \in A} P\{s'_t \mid s_t, \pi\} \cdot Q_\pi(s', a') \tag{27}$$

where s'_t and a'_t are the state and action, respectively, in time slot $t+1$. γ is the discount factor. R_t is the cumulative reward before time slot t.

The proposed MEC framework is designed to require the optimal offloading and bandwidth allocation policy π^* that incurs the minimum cost, namely, the maximum accumulative reward. Therefore, the optimal policy π^* is described by

$$\pi^* = \arg\max_{\pi} Q(s_t, a_t) \tag{28}$$

The process is thus an infinite-horizon MDP with a discount factor γ based on an agent that can be written as

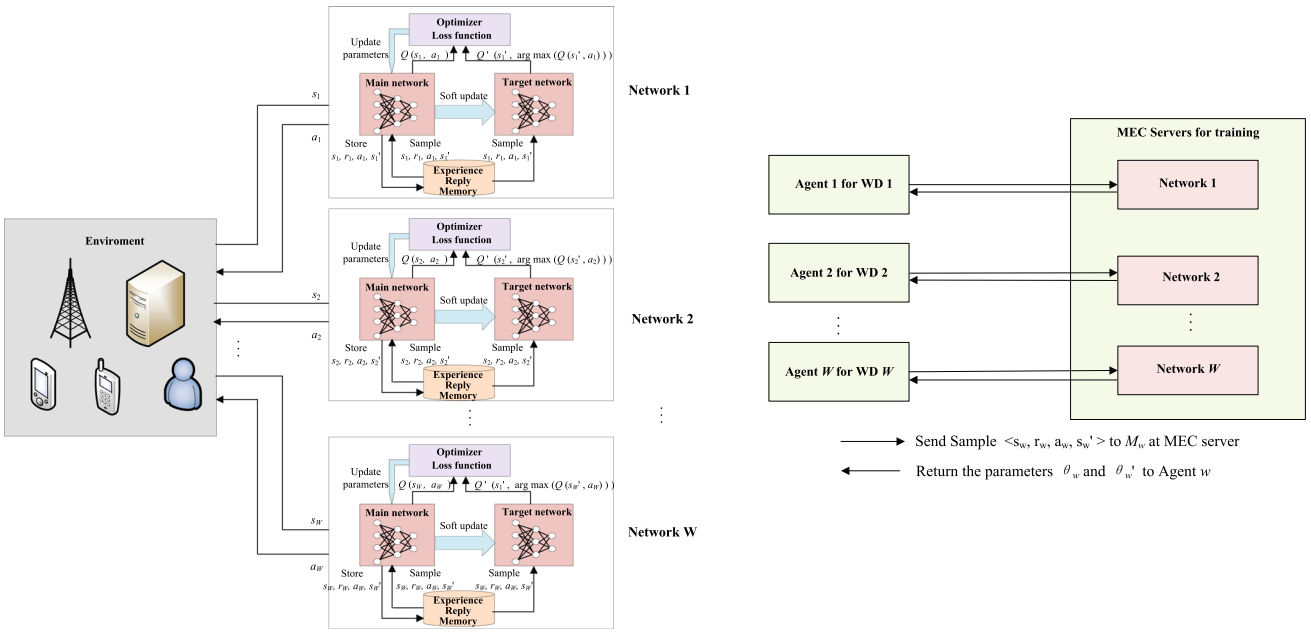
$$Q(\mathcal{S}, \mathcal{A}) = \mathbb{E}_{\pi^*}\left[\frac{1}{T} \cdot \lim_{t \to \infty} \sum_{t=1}^{T} Q(s'_t, a'_t) | s_t = s_0\right] \tag{29}$$

However, for the MDP model, the state must be known (from *priori* knowledge) when an agent observes from the proposed MEC environment, $P\{s'_t \mid s_t, \pi\}$ is the structure followed in the Markov chain. From (24), the state is composed of the channel state, SINR, and arriving computation task, which are unknown. More unfortunately, as the state and action space grow exponentially with the increasing of WD's number, which will give rise to the curse of dimensionality.

4.3 Optimal scheme based on DRL

We consider using DDQN, which is an off-policy and model-free DRL technique [39]. DDQN can separate the action selection from the action evaluation by various state-action functions to avoid overestimation of the DQN [40]. DRL combines DL and RL and constructs deep neural network to tackle the high-dimensional state and action space. For offloading and allocation optimization problem, we use DDQN to solve the high-dimensional state space when there are more WDs and propose a decentralized optimization scheme for partial offloading and bandwidth allocation based on DRL (DOCRRL). According to the neural network, DOCRRL can be used as estimation functions, and $Q(\mathcal{S}, \mathcal{A})$ is approximated as $Q(\mathcal{S}, \mathcal{A}|\theta)$. In terms of the proposed scheme, θ represents the parameters of the DOCRRL. The mentioned decentralized learning (decentralized DRL) means that each WD has a learning agent to interact with the proposed MEC environment, then observe the state independently, the immediate reward can be obtained. As long as the computation resources and training resources of MEC server-side are sufficient, the performance of decentralized DRL scheme will not decrease as the number of WDs increases.

In our proposed scenario, we need to find the optimal policy π^*, which consists of paired states and actions. By training the NNs, we can adaptively select the offloading ratio and the bandwidth allocation ratio according to the channel state, SINR, and arrival ratio of computation workloads to maximize the cumulative reward. To avoiding the curse of dimensionality arising from the massive WDs, the DOCRRL algorithm combined to deep neural network is proposed to solve the optimal scheme for our proposed MEC system. As shown in Fig. 3(a), there are W NNs that approximate $Q(\mathcal{S}, \mathcal{A}|\boldsymbol{\theta})$. without loss of



(a) Structure of DOCRRL **(b)** Decentralized training pattern

Fig. 3 Structure and training pattern of DOCRRL

generality, the state space $\mathcal{S}$ is divided into $\{s_1, s_2, \ldots, s_W\}$, and the action space $\mathcal{A}$ is divided into $\{a_1, a_2, \ldots, a_W\}$. Thus, $Q(\mathcal{S}, \mathcal{A})$ is denoted as $\{Q(s_1, a_1), Q(s_2, a_2), \ldots, Q(s_W, a_W)\}$. Two advanced techniques are exploited to improve the convergence of DOCRRL: fixing all parameters of the target network and experience replay memory. For WD w, the agent can observe from the environment which is our proposed MEC model; then state s_w can be obtained. From the main network, *Network w*, the agent chooses a_w according to the policy π_w, and the immediate reward r_w is generated; furthermore, the next state, s'_w, is generated from the target network. The parameter sets θ_w is updated by *Network w*. Furthermore, the main network, target network can be iteratively trained and replaced for minimizing the loss function and obtain the optimal policy. Here, the mean square error(MSE) is utilized to calculate the loss function,

$$\mathrm{L}(\theta_w) = (y_w - Q(s_w, a_w|\theta_w))^2 \tag{30}$$

where y is the object function, which is defined as

$$y_w = r_w + \max \gamma Q'\left(s'_w, \arg\max Q(s'_w, a_w|\theta_w), |(\theta'_w)\right) \tag{31}$$

In order to mitigate the computational pressure on the WDs side, the training processing of NNs is transferred to the MEC side. As shown in Fig. 3(b), there are W NNs in the proposed MEC framework. For our proposed DOCRRL algorithm, we let MEC servers help to train the NN for each WD w. This MEC server can be the node with the greatest remaining computation capacity or an inactive node that is not used to execute the offloaded workloads from WD[2]. It is reasonable to let MEC servers help with the training directly. This is because the information exchange involved in the training (including the state information and the parameters of NNs) is small. In addition, the required processing capacity of the training in each time slot can be much less than those of the workloads of WDs. Moreover, since the features of the input (the state s_w) of the NN for WD w are nearly same, these NNs can share parameters during training, which will greatly reduce the cost of training and alleviate the burden on the relevant MEC server. The mentioned experience replay memory M_w and two NNs (the main network and the target network) can be maintained by the MEC server of $m \in M$ for WD w.

The proposed DOCRRL algorithm are explained in Algorithm 1. When the WD w generates the computation workloads to be offloaded processing, WD sends a request to the MEC server m. If the parameters of NN for w need to be updated, the selected MEC server m forwards these parameters to WD w, WD w can perform the action a_w (the ratio of computation offloading and bandwidth resource allocation), then WD w can obtain the immediate reward r_w. To this end, the training sample sequence $< s_w, a_w, r_w, s'_w >$ can be obtained by combining the observed state from the environment. WD w sends the sample sequence $< s_w, a_w, r_w, s'_w >$ to

[2] The resources scheduling scheme for MEC servers that can be leveraged to train NNs is beyond the scope of this paper, so it is not explained here.

M_w of the MEC server m. At MEC server side, MEC server m receives sample sequence $< s_w, a_w, r_w, s'_w >$ from WD w and store sample sequence to M_w. To this end, DOCRRL randomly selects the mini-batch samples from M_w as training samples to train the main network, then obtains the loss and computes the gradient of the DOCRRL's main network. Finally, the parameters of DOCRRL's main network can be updated by stochastic gradient descent. If the number of iteration steps is greater than the threshold of maximum copy frequency C_{max}, copy all the parameters for the main network to the target network.

Algorithm 1 Training process of DOCRRL

Initialize the parameters for all WDs. The DOCRRL's main networks and target networks of WDs: $\boldsymbol{\theta} = \{\theta_1, \theta_2, \cdots, \theta_W\}, \boldsymbol{\theta'} = \{\theta'_1, \theta'_2, \cdots, \theta'_W\}$;
Initialize the parameters. The number of iterations is E_{max}, the frequency of the parameters for the target network is C_{max}, the learning rate is α, the exploration rate is ϵ, the maximum capacity of the experience replay memory is M_w, the time slots is T, the epochs is E, and the mini-batch is m_w,.
for episode=$[1, 2, \cdots, E_{max}]$ **do**
  Reset the environment of proposed MEC framework, initialize the agents;
  for $[t = 1, 2, \cdots, T]$ **do**
    if t mod $\lfloor T/E \rfloor = 0$
      Switch the encryption algorithm, active MEC servers;
    end if
    According to the interaction with the simulation environment, the agent can gain the initial state from $\mathcal{S}$;
    Reset the initial immediate reward;
    for $[w = 1, 2, \cdots, W]$ **do**
      if WD w requests the parameters of NNs(main DOCRRL network and target network)
        send request to corresponding MEC servers and receive θ_w and θ'_w;
        According to the policy π_w, select a_w by $\epsilon-greedy$;
        Obtain the immediate reward r_w. Furthermore, when agent w interacts with the MEC environment, observes the the next state s'_w;
      end if
      if MEC server receive the observe experience
        Observe experience replay memory M_w; if M_w is not full, push $< s_w, a_w, r_w, s'_w >$ into M_w on MEC servers;
      end if
      Randomly select the mini-batch m_w samples from M_w as training samples to train the DOCRRL's main network on MEC server;
      In line with (30) and (31), obtain the loss L and derive the gradient $\nabla_{\theta_w} L$ of the DOCRRL's main network;
      Update the parameters θ_w of the DOCRRL's main network using gradient descent;
    end for
    if the number of iteration steps is greater than the frequency C_{max}
      Copy all the parameters $\boldsymbol{\theta}$ of the DOCRRL's main network to parameters $\boldsymbol{\theta'}$ of the DOCRRL's target network;
    end if
  end for
end for

5 Numerical results

For evaluating the convergence and performance of the proposed DOCRRL algorithm, we implements extensive simulations with different parameter configurations and compares DOCRRL with four benchmark schemes.

5.1 Simulation setup

A personal computer with i9-9900K CPU with a lowest main frequency of 3.6 GHz and with 2080Ti video card with 11GB memory is used to train and test the proposed DOCRRL. The integrated development environment is Jetbrains Pycharm with Tensorflow 1.10.0 on Ubuntu 18.04 LTS.

In the simulations, we assume that there are W WDs scattered at equal distances of 100 m from the MBS. The MEC servers are connected to MBS by fiber. We set the number of WDs and MEC servers to five and four respectively; that is, $W = 5, M = 4$. We assume that the computation workloads generated by WDs in each time slot are the face recognition images[3]. The number of sub-workloads for each WD is divided into 10 levels. The number of time slots is set to $t = 1$ (1 delay unit equals 10 ms). The number of epochs E is set to 3, and all the epochs have the same duration. The maximum episodes E_{max} is 3000. Therefore, the duration of each epoch is 1000. The active MEC servers are MEC2 and MEC3 in epoch 1, MEC1 and MEC2 in epoch 2, and MEC2 and MEC4 in epoch 3. That is to say, during each epoch, two active MEC servers are set to process the offloaded computation workloads generated by 5 WDs. Moreover, $\rho_c = 0.95$, $\sigma^2 = 10^{-9}$, and $\kappa = 10^{-27}$. Furthermore, $c_w(t)$ is related to λ_w and ranges from $0.55 * 10^7$ to $2.19 * 10^7$ in eight steps, that is, $c_w(t) \in [0.55 * 10^7, 0.87 * 10^7, 1.14 * 10^7, 1.28 * 10^7, 1.60 * 10^7, 1.81 * 10^7, 2.00 * 10^7, 2.19 * 10^7]$. In addition, $T_{w,max} = 1$, $P_{l,w} = 2$ W, $P_{m,w} = 2$ W, $f_{max} = 3 * 10^8$, and $B_m = 5$ MHz. For the MEC servers, $\xi_1 = \{0.1, 0.2, 0.3, 0.4\}$, $\xi_2 = \{0.05, 0.1, 0.15, 0.2\}$, and $L_m = 10$. For each encryption algorithm, $\varsigma_1, \varsigma_2, \varsigma_3$ follow the uniform distribution with $U(0.1, 1)$, $\varsigma'_1 = 0.4, \varsigma'_2 = 0.5, \varsigma'_3 = 0.6$. In addition, $\lambda'_1 = 1.7, \lambda'_2 = 2.2$, and $\lambda'_3 = 3.0$.

In DOCRRL, all W NNs have the same structure, which consists of 2 layers fully connected networks. Furthermore, the main hyperparameters in the NNs of the DOCRRL algorithm are set as follows. The number of neurons is respectively 200 and 150. The capacity of experience replay memory M_w is set to 200,000. The learning rate α is 10^{-3} which will be discussed in Fig. 4 of Sect. 5.3. The

[3] We assume that 10 face recognition images is a batch of computation workloads and these workloads(1 batch of 10 face recognition images) follows the Poisson distribution with λ_w.

bound of action, state space are respectively 4 and 2. P_w is 10. And the bound of time slot T in one episode for training is set to 300. The size of the mini-batch m is set to 64. The copy frequency to the target networks C_{max} is set to 2048. The exploration rate ϵ is set to 0.9. The weights of the reward function are set to $\alpha_1 = 1/6, \alpha_2 = 1, \alpha_3 = 1, \alpha_4 = 1$, and $\alpha_5 = 1$.

5.2 Performance comparison

To evaluate and verify the DOCRRL's performance, four benchmark algorithms are introduced for comparison: the centralized optimization scheme on binary computation offloading and resource allocation based on deep reinforcement learning (COBRL), the greedy binary offloading policy (GBOP), offloading all workloads to MEC server (OTM), and local execution (LE) of all computation workloads.

We discretize the action space $\mathcal{A}$ into 11 levels. That is, the bound of $\mathcal{A}$, which consists of the ratio of offloaded computation workloads and the ratio of allocated bandwidth, is 1.0. The value of $\mathcal{A}$ is quantized from 0 to 1.0 in 0.1 increments.

(1) *COBRL*

Because the number of WDs is 5 in our proposed MEC framework which causes the centralized partial computation offloading DRL scheme to not converge, COBRL is designed to find the optimization scheme for binary offloading decision making. COBRL assume that all the workloads generated by WDs are indivisible, which means that all tasks scheduled by an agent are executed locally by WD's processors or offloaded to the MEC server for execution. For decrease the number of the actions for ensuring the convergence of COBRL, the action for the ratio of allocated bandwidth is discretized 3 levels with [0.3, 0.65, 1.0].

(2) *GBOP*

The GBOP determines whether to complete computation workloads by local execution or offloading to obtain the maximum cumulative reward. The execution model of all 5 WDs are scheduled by GBOP respectively. In current time slot t, if the cumulative reward obtained in local execution mode is greater than that in offload mode for workloads from each WD, local execution is selected for the certain WD in next time slot $t + 1$; otherwise, the offload mode is selected.

(3) *OTM*

Regardless of the size of workloads, the constraint on the bandwidth cost, all the workloads are forcibly offloaded to activated edge servers with the maximum transmission power and bandwidth.

(4) *LE*

All computation workloads are forcibly executed on the WD's processors with the maximum allocation power.

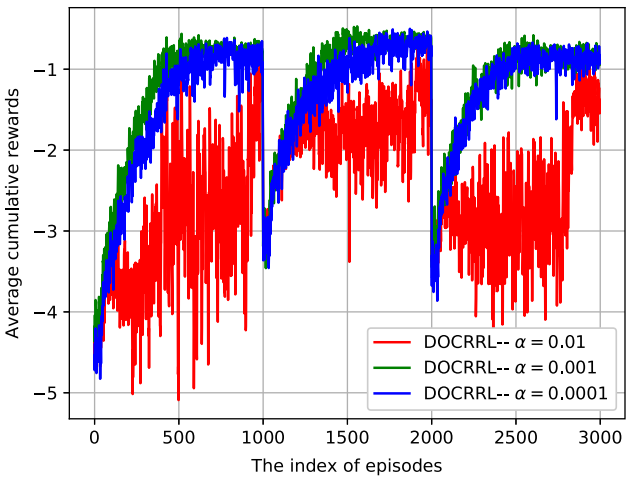


Fig. 4 Average cumulative reward for different learning rates

5.3 Simulation results

Taking into the convergence performance account, we verify the performance of DOCRRL at different learning rates α. Figure 4 shows three average cumulative reward curves for learning rates α of 0.01, 0.001, and 0.0001. For $\alpha = 0.01$ (red curve), the DOCRRL algorithm cannot converge because α is too large. For $\alpha = 0.001$ (green curve), DOCRRL obtains the optimal policy in epoch 1 after nearly 400 episodes. During epochs 2 and 3, DOCRRL converges after 350 episodes. For $\alpha = 0.0001$ (blue curve), DOCRRL can also obtain the optimal policy, but the convergence rate is lower than that for $\alpha = 0.001$. Therefore, we set $\alpha = 0.001$ to train the NN of the DOCRRL.

Figure 5 shows the ratio of offloading failure under different risk probabilities. For the three encryption algorithms, we choose the AES, RSA, and BGN algorithms, which provide symmetric encryption, asymmetric encryption, and homomorphic encryption, respectively. Although the constraint on the risk probability $P_{k,m}$ is the same for all three encryption algorithms, the performance is different. The ratio of offloading failure for the AES algorithm is strongly affected by the risk probability $P_{k,m}$. As $P_{k,m}$ increases, the offloading failure ratio for AES dramatically decreases. We can see that when $P_{k,m} \leq 0.5$, the three algorithms have higher offloading failure ratios, because the lower threshold of the risk probability strongly affects

the performance of encryption algorithms. By contrast, when $P_{k,m} > 0.5$, the AES, RSA, and BGN algorithms work well and are not significantly affected by the risk probability constraint, especially for $P_{k,m} = 0.7$. To improve the offloading efficiency, we set the threshold of the risk probability to $P_{k,m} = 0.6$.

We will describe in detail the performance of our proposed DOCRRL scheme in different environments for different values of the adjustment coefficient.

Figure 6 shows the average cumulative reward for DOCRRL compared to those of the other four algorithms in epochs 1, 2, and 3. We set the size of the arriving computation workloads for each WD λ_w to 5; then $\mathbb{E}[c_w(t)] = 1.60 * 10^7$. To balance the costs of the latency, power, and bandwidth, the adjustment coefficients ω_1, ω_2 are set to 0.5. That is, the penalty relative to the cost of the total delay is equal to energy consumption and available bandwidth. We can see that the average cumulative rewards of DOCRRL in the three epochs are larger than those of the other four algorithms. Although DOCRRL's rewards are not great in the initial episodes, DOCRRL can gradually learn the optimal offloading and bandwidth allocation policy during the training process. DOCRRL outperforms the COBRL, GBOP, OTM, and LE owing to its use of decentralized learning. The COBRL and GBOP determine whether to complete computation workloads by local execution or computation offloading for obtaining the maximum accumulative rewards whose performance is better and stable. OTM offloads all the computation workloads without considering the size of workloads or the available bandwidth. Because the computation workloads are not very large and the bandwidth can meet the requirements of all the WDs, the performance of the OTM is not greatly affected. For the LE, the computation workloads are too large for all the WDs and cannot be executed by the WD processors under the deadline constraint. The penalty P_w for the failure to complete workloads is very large, and smaller cumulative rewards are obtained, so the performance of LE is degraded dramatically.

Figure 7 compares the performance of the five algorithms at different workloads arrival rates. We set the parameter ω_1 to 0.5. Moreover, the range on the workload arrival rate for each WD λ_w is set to [1, 8]. Because the DOCRRL algorithm can use decentralized learning for obtaining the optimal offloading and bandwidth allocation policy for each WD, and the number of WDs does not affect its performance when the bandwidth is sufficient. As shown in Fig. 7(a), DOCRRL's average cumulative rewards are greater than those of the other four algorithms regardless of the arrival rate of workloads. Owing to decentralized processing for DOCRRL scheme, as the workload arrival rate increases, DOCRRL is much better than those of four benchmarks. Figure 7(b) illustrates the total latency of local execution for five algorithms. LE has the worst performance among the algorithms because the processing power of the WDs is much lower than that of the MEC servers. Most of the workloads do not meet the requirement of local execution. The total delay of local execution for OTM is 0 because all of the workloads are selected to offloading execution. As the workload arrival rate increases, more workloads is scheduled to offload execution on MEC servers, so the local delay of COBRL gradually decreases. DOCRRL and GBOP exhibit very similar performance. As shown in Fig. 7(c), the results for the total delay of offloading are similar to those for the total delay of local execution. The average delay of offloading for LE is 0 because all the workloads are offloaded to the MEC server. The DOCRRL algorithm outperforms the COBRL, GBOP and OTM because of its efficient learning process. Figure 7(d) shows the total power consumption for

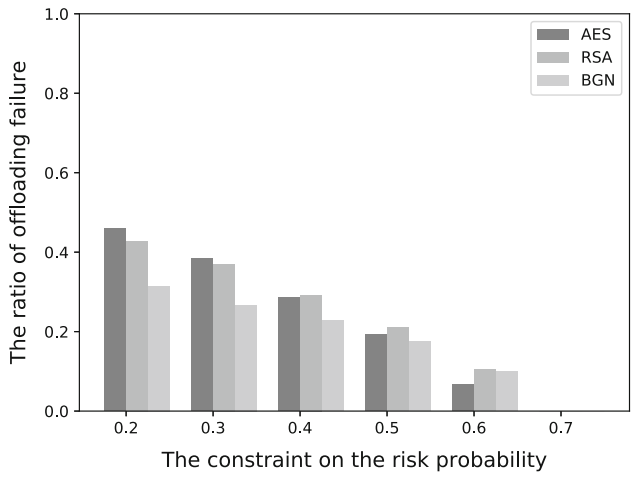


Fig. 5 Comparison of the risk probabilities

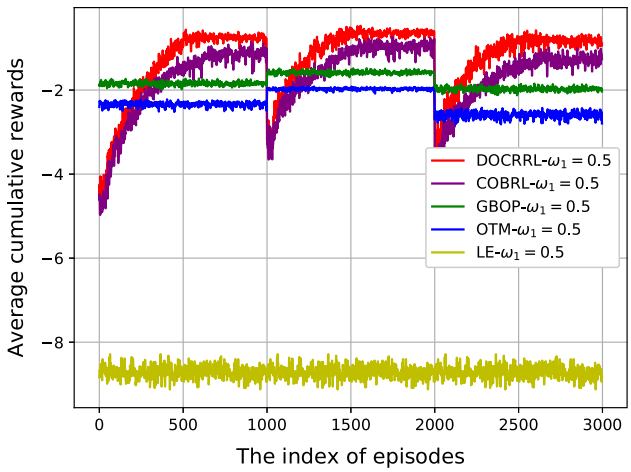


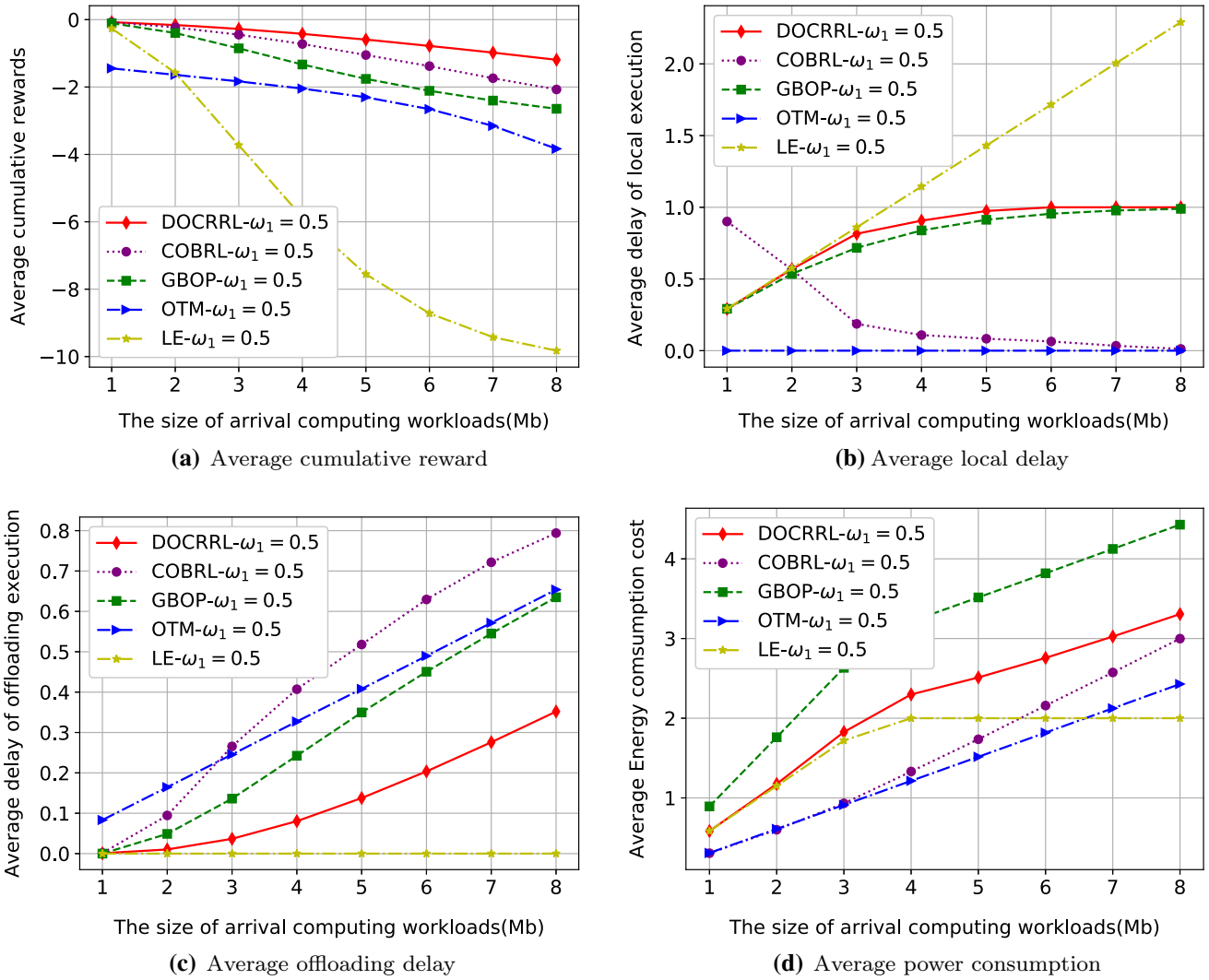
Fig. 6 Average cumulative rewards obtained using different algorithms for $\omega_1 = 0.5$

local execution and offloading for the five algorithms. Although the total power consumption for DOCRRL is slightly higher than those of COBRL, OTM and LE, COBRL can adjust the ratio of bandwidth allocation, OTM uses more of the resources of the MEC servers, and LE is affected by the penalty for the failure to complete workloads. Considering the average cumulative rewards, DOCRRL performs better than other four algorithms.

The action space $\mathcal{A}$ consists of the computation workload offloading ratio and the bandwidth allocation ratio. We examine these ratios for the five algorithms in Fig. 8 and Fig. 9. We set the parameter ω_1 to 0.5. The total workload arrival rate for each WD λ_w is in the range [1, 8], which is same as the settings in Fig. 7.

Figure 8 shows the average offloading ratios of the DOCRRL, COBRL, GBOP, OTM, and LE. OTM selects to offload all of the workloads to activated MEC server, but the performance is affected by the risk probability threshold, and the offloading ratio is not 1.0. In contrast to that of OTM, the offloading ratio of LE is 0 because all the computation workloads are executed locally by the WDs. Owing to its use of decentralized processing, the offloading ratio of the DOCRRL algorithm is lower than that of the GBOP. The DOCRRL algorithm reduces the cost of the MBS and bandwidth allocation as much as possible. The COBRL selects the binary offload mode for all computation workloads, and the average offloading ratio of the COBRL indicates the proportion of offloaded workloads in one episode. Thus, when the arrival rate of computation workloads increases, the average offloading ratio for COBRL increases rapidly compared to that of the DOCRRL algorithm. GBOP's workload offloading ratio is less than that of COBRL, because GBOP can select local execution or offload computation workloads for each WD.

Figure 9 shows the bandwidth allocation ratio versus the size of the arriving computation workload for the five



(a) Average cumulative reward

(b) Average local delay

(c) Average offloading delay

(d) Average power consumption

Fig. 7 Comparison of the performance of five algorithms at different task arrival rates

algorithms. It is easy to see that the bandwidth allocation ratio for LE is 0 because all of the workloads are forced to local execution. OTM selects maximum bandwidth to offload workloads; therefore, the bandwidth allocation ratio is always 1. When the rate of arrival workloads is greater than 6, all WDs do not have sufficient computation resources, and most workloads are chosen to offload to activate edge servers because of the greater bandwidth allocation ratio. The bandwidth allocation ratios of DOCRRL COBRL, GBOP are close to 1.

Figure 10 shows the performance for DOCRRL compared to those of the other four algorithms for different weights ω_1. The size of the computation workloads λ_w is set to 5; therefore, $c_w(t) = 1.60 * 10^7$. The range of ω_1 is set to [0.1, 0.9]. As ω_1 increases, the average cumulative rewards of the five algorithms gradually decrease. The reason is that the power consumption and bandwidth penalties become larger. DOCRRL outperforms COBRL, GBOP, OTM, and LE no matter what value of ω_1 is. Owing to the insufficient processing ability of WDs, the LE method was penalized for failure to process workloads; therefore, it had the worst performance.

Figure 11 illustrates the performance of DOCRRL compared to four benchmarks for different local execution delay coefficients α_1. The size of the computation workloads λ_w is set to 5; therefore, $c_w(t) = 1.60 * 10^7$. The range of α_1 for each WD is set to [1/6, 1/5, 1/4, 1/3, 1/2, 1]. Because α_1 ranges from 1/6 to 1, the penalty for local execution increases gradually. The average cumulative rewards of the five algorithms decrease. When $\alpha_1 = 1/2$ or 1, DOCRRL, COBRL, GBOP, and OTM exhibit very similar performance. The average cumulative reward of OTM does not change with α_1, because the experiment considered only the local delay coefficient α_1, which does not affect the computation offloading.

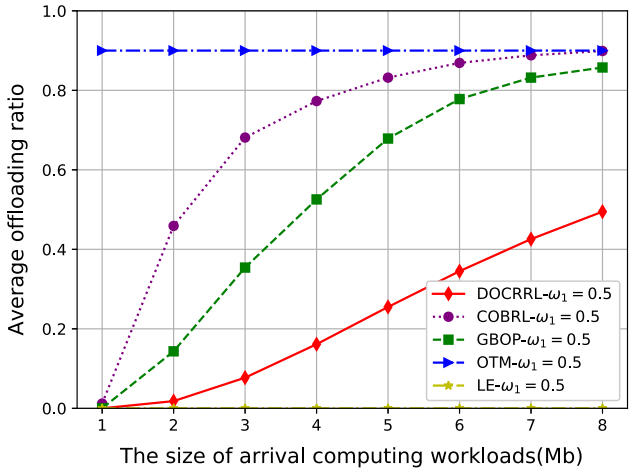


Fig. 8 Comparison of the offloading ratios of the five algorithms

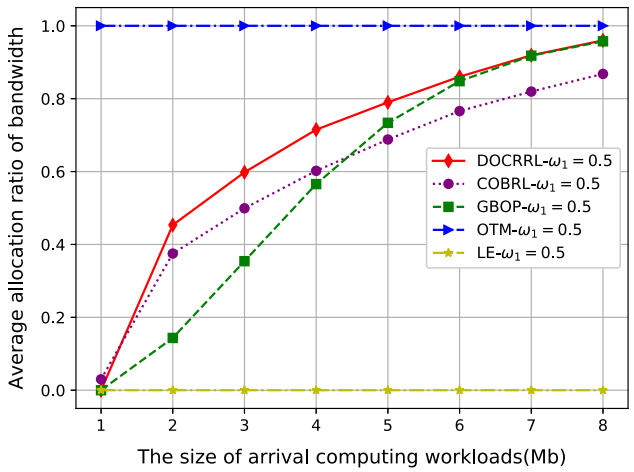


Fig. 9 Comparison of the allocated bandwidth for the five algorithms

6 Conclusions

We implemented an MEC model with multiple WDs and MEC servers. To obtain the minimum weighted sum cost, we proposed DOCRRL, which is a DRL-based decentralized optimization scheme for partial computation offloading and bandwidth allocation. Taking into the time-varying channel condition, the stochastic arrival of workloads accounting, the DOCRRL algorithm can adaptively select the offload mode and bandwidth ratio that yield the optimal policy under the delay and risk probability constraints. Extensive simulation results reveal that DOCRRL set out the stable learning ability and powerful performance for various parameter configurations. Furthermore, we consider optimizing the multi-base station selection problem, and apply federated learning[41] to tackle computation offloading and privacy protection decision-making problem in the MEC scenario, which can be more meaningful work plan.

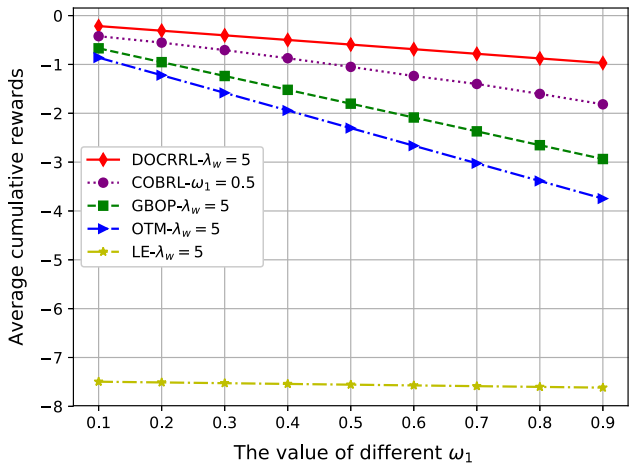


Fig. 10 Comparison results with different values of ω_1

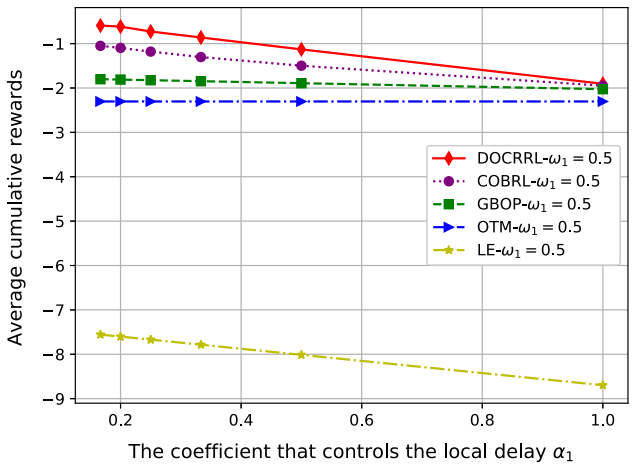


Fig. 11 Comparison of the performance of the five algorithms for different values of α_1

Acknowledgements This work was supported by the National Nature Science Foundation of China (61841602, 61806024), the Jilin Province Education Department Scientific Research Planning Foundation of China (JJKH20210753KJ, JJKH20200618KJ), and the Jilin Province Science and Technology Department Project of China (20190302106GX).

References

1. Kiran, N., Pan, C., Wang, S., & Yin, C. (2020). Joint resource allocation and computation offloading in mobile edge computing for sdn based wireless networks. *Journal of Communications and Networks, 22*(1), 1–11.
2. Ahmadabadian, M., Moghaddam, S. K., & Razavizadeh, S. M. (2020). Energy efficiency maximization in fdd massive mimo systems with channel aging. *Wireless Networks, 26*(2).
3. Park, S., Kang, Y., Tian, Y., & Kim, J. (2020) Fast and reliable offloading via deep reinforcement learning for mobile edge video computing. In: 2020 International Conference on Information Networking (ICOIN), pp. 10–12
4. Hadded, L., Charrada, F.B., & Tata, S. (2018). Efficient resource allocation for autonomic service-based applications in the cloud. In: 2018 IEEE International Conference on Autonomic Computing (ICAC), pp. 193–198. IEEE
5. Du, J., Zhao, L., Feng, J., & Chu, X. (2018). Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications, 66*(4), 1594–1608.
6. Liu, Y., Xiong, K., Ni, Q., Fan, P., & Letaief, K. B. (2020). Uav-assisted wireless powered cooperative mobile edge computing: Joint offloading, cpu control, and trajectory optimization. *IEEE Internet of Things Journal, 7*(4), 2777–2790. https://doi.org/10.1109/JIOT.2019.2958975.
7. Zhou, F., & Hu, R. Q. (2020). Computation efficiency maximization in wireless-powered mobile edge computing networks. *IEEE Transactions on Wireless Communications, 19*(5), 3170–3184.
8. Sun, Y., Xu, L., Tang, Y., & Zhuang, W. (2018). Traffic offloading for online video service in vehicular networks: A cooperative approach. *IEEE Transactions on Vehicular Technology, 67*(8), 7630–7642.
9. Pasha, M., & Rahman Khan, K. U. (2018). Scalable and energy efficient task offloading schemes for vehicular cloud computing. *International Journal of Computer Networks and Communications (IJCNC), 10,*.
10. Zhan, W., Luo, C., Wang, J., Wang, C., Min, G., Duan, H., & Zhu, Q. (2020). Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing. *IEEE Internet of Things Journal, 7*(6), 5449–5465.
11. Sun, F., Hou, F., Cheng, N., Wang, M., Zhou, H., Gui, L., & Shen, X. (2018). Cooperative task scheduling for computation offloading in vehicular cloud. *IEEE Transactions on Vehicular Technology, 67*(11), 11049–11061.
12. Liu, J., Kumar, K., Lu, Y.-H.: Tradeoff between energy savings and privacy protection in computation offloading. In: 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED), pp. 213–218 (2010). IEEE
13. Liu, J., Lu, Y.-H.: Energy savings in privacy-preserving computation offloading with protection by homomorphic encryption. In: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, HotPower, vol. 10, pp. 1–7 (2010)
14. Goudarzi, M., Zamani, M., & Haghighat, A. T. (2017). A fast hybrid multi-site computation offloading for mobile cloud computing. *Journal of Network and Computer Applications, 80,* 219–231.
15. Shahzad, H., & Szymanski, T.H. (2016). A dynamic programming offloading algorithm for mobile cloud computing. In: 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1–5. IEEE
16. Valentino, R., Jung, W.-S., & Ko, Y.-B. (2018). A design and simulation of the opportunistic computation offloading with learning-based prediction for unmanned aerial vehicle (uav) clustering networks. *Sensors, 18*(11), 3751.
17. Lyu, X., Tian, H., Jiang, L., Vinel, A., Maharjan, S., Gjessing, S., & Zhang, Y. (2018). Selective offloading in mobile edge computing for the green internet of things. *IEEE Network, 32*(1), 54–60.
18. Ning, Z., Dong, P., Kong, X., & Xia, F. (2018). A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things. IEEE Internet of Things Journal
19. Tang, Q., Xie, R., Yu, F. R., Huang, T., & Liu, Y. (2020). Decentralized computation offloading in iot fog computing system with energy harvesting: A dec-pomdp approach. *IEEE Internet of Things Journal, 7*(6), 4898–4911. https://doi.org/10.1109/JIOT.2020.2971323.
20. Wang, C., Liang, C., Yu, F. R., Chen, Q., & Tang, L. (2017). Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Transactions on Wireless Communications, 16*(8), 4924–4938.
21. Qian, L. P., Feng, A., Huang, Y., Wu, Y., Ji, B., & Shi, Z. (2018). Optimal sic ordering and computation resource allocation in mec-aware noma nb-iot networks. *IEEE Internet of Things Journal, 6*(2), 2806–2816.
22. Tan, Z., Yu, F.R., Li, X., Ji, H., & Leung, V.C. (2017). Virtual resource allocation for heterogeneous services in full duplex-enabled small cell networks with cache and mec. In: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 163–168 . IEEE
23. Wu, X., Jiang, W., Zhang, Y., & Yu, W. (2019). Online combinatorial based mechanism for mec network resource allocation. *International Journal of Communication Systems, 32*(7), 3928.
24. Wang, J., Hu, J., Min, G., Zhan, W., Ni, Q., & Georgalas, N. (2019). Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning. *IEEE Communications Magazine, 57*(5), 64–69.

25. Ning, Z., Dong, P., Wang, X., Guo, L., Rodrigues, J. J., Kong, X., et al. (2019). Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational offloading scheme. *IEEE Transactions on Cognitive Communications and Networking, 5*(4), 1060–1072.
26. Liu, Y., Yu, H., Xie, S., & Zhang, Y. (2019). Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology, 68*(11), 11158–11168.
27. Huang, L., Bi, S., & Zhang, Y. J. (2019). Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Transactions on Mobile Computing*.
28. Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y., & Bennis, M. (2018). Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet of Things Journal
29. Wang, Y., Wang, K., Huang, H., Miyazaki, T., & Guo, S. (2018). Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. *IEEE Transactions on Industrial Informatics, 15*(2), 976–986.
30. Xiao, L., Li, Y., Huang, X., & Du, X. (2017). Cloud-based malware detection game for mobile devices with offloading. *IEEE Transactions on Mobile Computing, 16*(10), 2742–2750.
31. He, X., Liu, J., Jin, R., Dai, H. (2017). Privacy-aware offloading in mobile-edge computing. In: GLOBECOM 2017-2017 IEEE Global Communications Conference, pp. 1–6. IEEE
32. Min, M., Wan, X., Xiao, L., Chen, Y., Xia, M., Wu, D., & Dai, H. (2018). Learning-based privacy-aware offloading for healthcare iot with energy harvesting. *IEEE Internet of Things Journal, 6*(3), 4307–4316.
33. Elgendy, I. A., Zhang, W., Tian, Y.-C., & Li, K. (2019). Resource allocation and computation offloading with data security for mobile edge computing. *Future Generation Computer Systems, 100,* 531–541.
34. Ngo, H. Q., Larsson, E. G., & Marzetta, T. L. (2013). Energy and spectral efficiency of very large multiuser mimo systems. *IEEE Transactions on Communications, 61*(4), 1436–1449.
35. Suraweera, H. A., Tsiftsis, T. A., Karagiannidis, G. K., & Nallanathan, A. (2011). Effect of feedback delay on amplify-and-forward relay networks with beamforming. *IEEE Transactions on Vehicular Technology, 60*(3), 1265–1271.
36. Ke, H., Wang, J., Wang, H., & Ge, Y. (2019). Joint optimization of data offloading and resource allocation with renewable energy aware for iot devices: A deep reinforcement learning approach. *IEEE Access, 7,* 179349–179363.
37. Sutton, R.S., Barto, A.G., et al. (1998). Introduction to Reinforcement Learning vol. 135. MIT press.
38. Sutton, R.S., McAllester, D.A., Singh, S.P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1057–1063
39. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.
40. Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In: Thirtieth AAAI Conference on Artificial Intelligence
41. Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST), 10*(2), 1–19.

H. C. Ke received his BS, MS degree at college of computer science and technology fromJilin University, respectively in 2004, 2007. He is interested in topics related to wireless networksand vehicular networks, especially for offloading and caching optimization. Currently he is anassociate Professor in Changchun Institute of Technology and studying Computer ApplicationTechnology at Jilin University to get his Ph.D. degree. He has published more than 20 papers oninternational publications.

H. Wang received her BS, MS, and Ph.D. degrees in college of computer science andtechnology from Jilin University, respectively in 2004, 2007, and 2010. Now she is interested intopics related to wireless networks and machine learning. Currently she is worked in ChangchunUniversity of Technology and has published more than 20 papers on international publications.

H. W. Zhao received his BS, MS degrees in college of computer science and technology, hisPh.D. degree in college of mechanical and aerospace engineering from Jilin Universityrespectively in 1986, and 1996, 2000. He is interested in topics related to intelligent informationsystem and wireless network. He has published over 30 articles on international journals.Currently he is a professor in Jilin University, China.

W. J. Sun received his BS degrees in college of computer science and technology from LiaoningNormal University, and MS degrees in college of computer science and technology from JilinUniversity, respectively in 1990, and 2004. He is interested in topics related to wirelesscommunication and network security. Currently he is a professor in Changchun University ofTechnology. He has published more than 30 papers on international publications.