DOI: 10.19884/j.1672-5220.202401003

Computation Rate Maximization for Wireless-Powered and Multiple-User MEC System with Buffer Queue

ABDUL Rauf*, ZHAO Ping
College of Information Science and Technology, Donghua University, Shanghai 201620, China

Abstract: Mobile edge computing (MEC) has a vital role in various delay-sensitive applications. With the increasing popularity of low-computing-capability Internet of Things (IoT) devices in industry 4.0 technology, MEC also facilitates wireless power transfer, enhancing efficiency and sustainability for these devices. The most related studies concerning the computation rate in MEC are based on the coordinate descent method, the alternating direction method of multipliers (ADMMs) and Lyapunov optimization. Nevertheless, these studies do not consider the buffer queue size. This research work concerns the computation rate maximization for wireless-powered and multiple-user MEC systems, specifically focusing on the computation rate of end devices and managing the task buffer queue before computation at the terminal devices. A deep reinforcement learning (RL)-based task offloading algorithm is proposed to maximize the computation rate of end devices and minimizes the buffer queue size at the terminal devices. The central idea of this work is to explore the best optimal mode selection for IoT devices connected to the MEC system. The proposed algorithm optimizes computation delay by maximizing the computation rate of end devices and minimizing the buffer queue size before computation at the terminal devices. Then, the current study presents a deep RL-based task offloading algorithm to solve such a mixed-integer and non-convex optimization problem, aiming to get a better trade-off between the buffer queue size and the computation rate. The extensive simulation results reveal that the presented algorithm is much more efficient than the existing work to maintain a small buffer queue for terminal devices while simultaneously achieving a high-level computation rate.

Key words: computation rate; mobile edge computing (MEC); buffer queue; non-convex optimization; deep reinforcement learning
CLC number: TN929.5 **Document code**: A
Article ID: 1672-5220(2024)06-0689-13

Open Science Identity (OSID)

0 Introduction

Currently, there are 13.8 billion Internet of Things (IoT) devices worldwide, which will reach 30.9 billion in 2025[1]. Meanwhile, these IoT devices have limitations regarding latency-sensitive applications and services, such as the Internet of Vehicles, face recognition, augmented reality and interactive gaming[2-6]. In such a circumstance, mobile edge computing (MEC) has arisen and received considerable attention[5-6]. This work concerns computation rate maximization for a wireless-powered and multiple-user MEC system with the buffer queue.

MEC provides an IT service experience at a place regarded to be a beneficial point in the mobile network system, commonly known as the radio access network (RAN) edge. This low-latency, close proximity and high-bandwidth environment offers localized cloud computing capabilities and simultaneously provides access to real-time context information and RAN[7]. The benefits of utilizing MEC with IoT devices are reduced latency, saving transmission cost and lowering entry barriers, which motivates us to continue with this research study.

MEC servers have another key feature that is powered with green energy or renewable energy resources, i.e., solar, wind or a geothermal resource that is readily available in the locality[8]. MEC servers also employ wireless-powered transmission to IoT devices linked to the network. MEC servers are, therefore, a primary incentive in this research.

A wireless-powered MEC server handles radio frequency (RF) energy transmission and accepts computational tasks from wireless user devices (WDs). An illustration of a wireless-powered and multiple-user MEC system is given in Fig. 1. Each WD connected with the MEC system is well equipped with a battery, a small computation unit and a communication circuit, i.e., a

Received date: 2024-01-10
Foundation items: National Natural Science Foundation of China (No. 61902060); Shanghai Sailing Program, China (No. 19YF1402100); Fundamental Research Funds for the Central Universities, China (No. 2232019D3-51); Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications, China) (No. SKLNST-2021-1-06)
* Correspondence should be addressed to ABDUL Rauf, email: a.rauf312@gmail.com
Citation: ABDUL R, ZHAO P. Computation rate maximization for wireless-powered and multiple-user MEC system with buffer queue[J]. *Journal of Donghua University (English Edition)*, 2024, 41(6): 689-701.

single antenna. A WD in this framework could range from low-power IoT systems, like environmental monitoring sensors strategically placed throughout the city, to the components of a dynamic smart security network, which encompasses both stationary surveillance equipment and mobile units for real-time public safety management. The MEC system is capable of broadcasting RF energy to the linked WDs to the network. In terms of computation resources, the MEC server is more powerful than the linked WDs. Therefore, the WDs can offload their computation tasks to the MEC server so that they can save their energy resources and get benefit from the MEC system's computation power. Each WD in the network is equipped with a rechargeable battery capable of storing the energy gathered from the MEC server. The accumulated energy powers the operations of each WD.

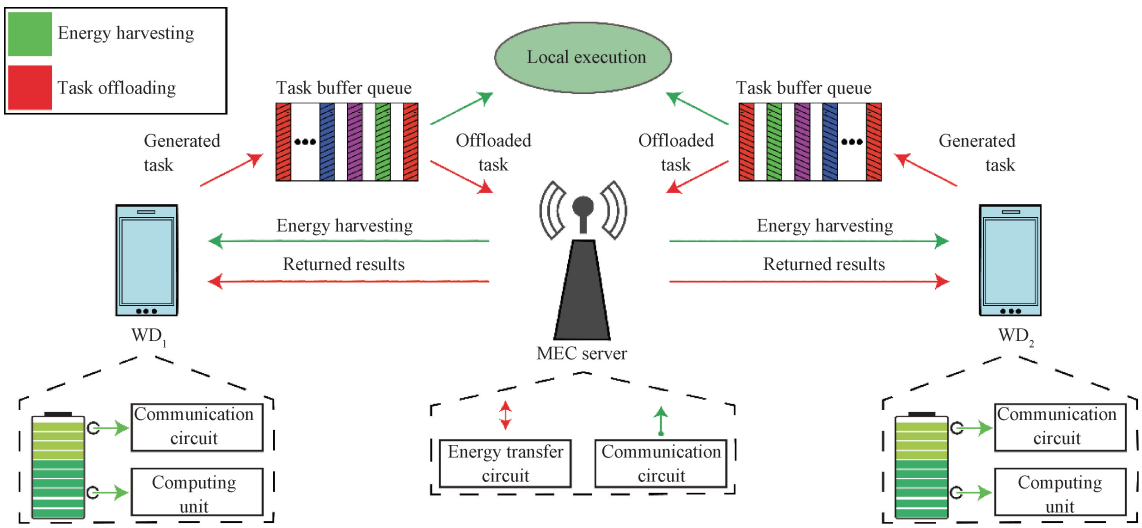

Fig. 1 Illustration of wireless-powered and multiple-user MEC system with buffer queue

A basic research challenge in the conventional battery-powered MEC system is the collaborative design of system resource allocation and task offloading. Similarly, many key factors constrain the computation performance in a wireless-powered and multiple-user MEC system with a buffer queue. Specifically, the availability of a free channel for task offloading plays a significant role[9].

In addition, the buffer queue size increasing with the task accumulation will significantly affect the computation rate and further affect the time delay of the whole MEC system. Furthermore, the energy harvested by the WDs affects the computation capacity of devices, and thus the computation rate. Indeed, it is essential to design an optimal offloading mechanism that takes the above factors into consideration and recommends the best optimal offloading decision to the WDs. Consequently, the best optimal offloading decision makes the computation rate at its optimized maximum.

Optimal mode selection provides the best usage of resources at a specific time frame, thereby significantly reducing both the execution latency and workload of the size-constrained IoT devices[10]. Utilizing optimal mode selection for offloading WD tasks makes the best use of the computation capabilities of the MEC system. In this research work, deep reinforcement learning (RL) is a promising method for improving MEC performance by minimizing the buffer queue size and maximizing the computation rate while also considering energy harvesting and channel gain. It can predict optimal mode selection based on given circumstances in a certain time frame.

Many existing studies[11-13] concerning the computation rate maximization in the MEC system are based on traditional optimization techniques, e. g., the alternating direction method of multipliers (ADMMs), the coordinate descent method and Lyapunov optimization. Nevertheless, these works need a considerable number of iterations to reach local optimum and thus are not applicable to real-time offloading decisions in fast fading channels. Another kind of related studies[14-18] is based on machine learning and focused on task offloading in the MEC system. However, these studies do not take the buffer queue (i. e., workload management) into consideration.

The current research investigates the computation rate maximization problem for a wireless-powered and multiple-user MEC system with a buffer queue. It presents an approach of task offloading to the MEC server based on deep RL that optimizes the weighted sum of WDs' computation rates. At the same time, it helps to keep the WD's buffer queue size to a minimum. A wireless-powered MEC system is shown, in which the MEC server transfers energy to WDs wirelessly and receives the offloaded tasks from these WDs on the basis of offloading decisions recommended by the algorithm presented in this research work. Thus, the main idea is to collaboratively optimize the time-allocation between transmitting energy and offloading tasks from each WD and the size of the buffer queue of all WDs. Therefore, for the MEC system with several users, it is necessary to

make an optimal offloading decision. The overall contributions are summarized as follows.

1) This research work formalizes the task offloading problem as an optimization problem that maximizes the computation rate and minimizes the buffer queue size via considering the channel gain, the buffer queue size and the wireless power transfer (WPT).

2) This research work proposes a deep RL-based task offloading algorithm for solving such a non-convex and mixed-integer optimization problem, aiming to get a better trade-off between the computation rate and the buffer queue size.

3) Extensive simulation results demonstrate that the deep RL-based task offloading algorithm is much more efficient than the existing work to maintain the buffer queue size at a low level while achieving a high computation rate at the same time.

The rest of the paper is organized as follows. In Section 1, we formalize the optimization problem for task offloading. Thereafter, we design a deep RL-based task offloading algorithm to solve such an optimization problem in Section 2, followed by the performance evaluation in Section 3. Finally, we conclude the whole paper in Section 4.

1 Formalizing Optimization Problem for Task Offloading in MEC System

1.1 System model

A wireless-powered MEC server handles RF energy transmission and accepts computational tasks from m WDs. The set of all WDs connected to the MEC server is denoted by $M = \{1, 2, \cdots, m\}$. Each WD is equipped with a communication circuit, i.e., a single antenna. It might be a low-power IoT system, such as a static sensors system or a smart security network.

The MEC server has a reliable power supply source powered by green energy, i.e., solar, wind and a geothermal energy source, and is capable of broadcasting RF energy to the linked WDs. The MEC server has more computing resources than the WDs. As a result, the WDs may offload their tasks to the MEC server, which can save their energy resources. Each WD is equipped with a rechargeable battery capable of storing the RF energy gathered from the MEC server. Each WD's operations are powered by the harvested energy from the MEC server. WPT and the communications are adopted to occur in the same frequency band. Adaptationally, a time-division-multiplexing (TDM) circuit is considered to be installed in each WD of the MEC system. TDM circuits prevent mutual interference between communication and WPT.

1.2 System time

The system time is divided into T time frames, and each time frame is equal-length and less than the channel coherence time, e.g., on the order of a few seconds. At each time frame, the wireless channel gain is proportional to both the communication speed and the amount of the harvested energy by a WD. The channel in both down-link and up-link is considered to be reciprocal and remains unchanged in each time frame, but it may vary among different time frames[19-20].

The MEC server initially allocates eT time in each time frame, where $e \in [0, 1]$ and is the time that the MEC server broadcasts RF energy towards WDs. So they can harvest energy for their needs. $\tau_i T$ is the time during which the MEC server receives the tasks offloaded from WDs. Assume that the offloading time for a specific WD who does not offload tasks to the MEC server is 0. Then, the time constraint is

$$e + \sum_{i=0}^{M} \tau_i \leqslant 1. \tag{1}$$

Furthermore, the duration of a single offloading time frame τ is separated into two segments, one for offloading tasks and the other for computing tasks. To guarantee that the assigned tasks can be accomplished in a single time frame, we assume that the total execution time of tasks remains constant and a larger task is divided into several smaller sub-tasks for execution. The illustration of a single time frame is given in Fig. 2.

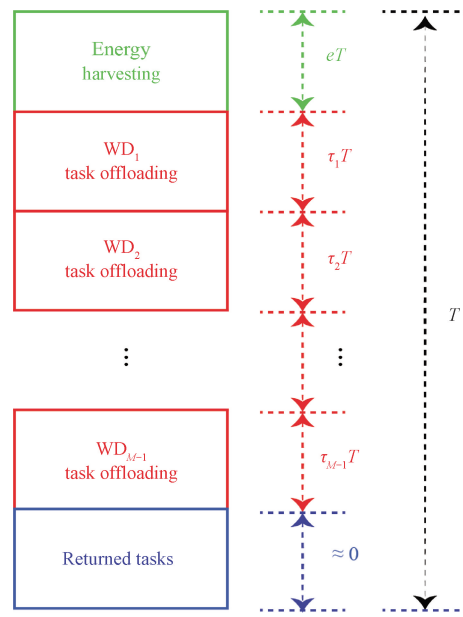

Fig. 2 Illustration of a single time frame

1.3 Data transmission and queuing model

As shown in Fig. 3, we consider that each WD runs one application. Each WD has one task buffer queue $B_i(t)$ where tasks are stored before assigning to local execution or offloading to the MEC server. Arrived application tasks are first stored in task queues, and wait for the task assignment. Each WD receives the computing results of the task execution immediately when the MEC server returns results to the WDs. In each time frame, the

arrived task $A_i(t)$ in bits generated by the ith WD is considered as independent and identically distributed. The set of arrived tasks from different WDs is represented as $A(t) = \{A_1(t), A_2(t), \cdots, A_i(t), \cdots, A_M(t)\}$. Likewise, the arrived task from the ith application in the time frame t will be stored in the task buffer queue $B_i(t)$ where the tasks await for their assignments. $B_i(t)$ in bits denotes the buffer queue size of the ith WD. The set of tasks in the buffer queue is denoted by $B(t)$:

$$B(t) = \{B_1(t), B_2(t), \cdots, B_i(t), \cdots, B_M(t)\}. \quad (2)$$

Note that $|B_i(t)| = |A_i(t)|$. Additionally, in a certain time frame t, we refer to the size of the data executed locally by the ith WD as $D_{i,\mathrm{L}}(t)$, and denote the size of the data computed by the MEC server by $D_{i,\mathrm{E}}(t)$.

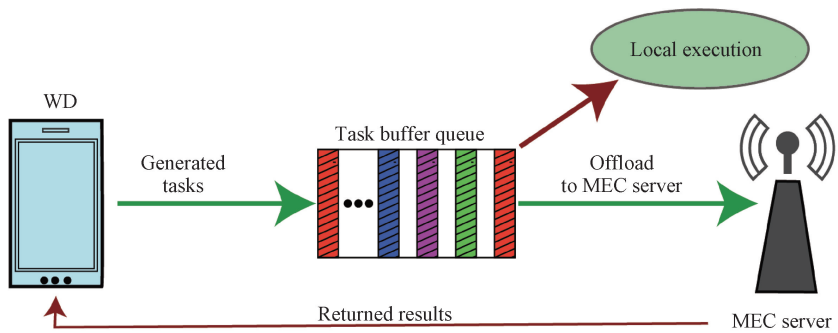

Fig. 3 Data transmission and queuing model for a WD

A mode selection parameter x_i is defined for the ith WD. $x_i = 0$, when the ith WD locally computes its tasks. Similarly, $x_i = 1$, when the ith WD offloads its tasks to the MEC server. The size of the tasks sent to the MEC server and the size of the tasks locally executed are defined as $D_{\mathrm{E}}(t)$ and $D_{\mathrm{L}}(t)$, respectively.

$$D_{\mathrm{E}}(t) = \{D_{1,\mathrm{E}}(t), D_{2,\mathrm{E}}(t), \cdots, D_{M,\mathrm{E}}(t)\}. \quad (3)$$

$$D_{\mathrm{L}}(t) = \{D_{1,\mathrm{L}}(t), D_{2,\mathrm{L}}(t), \cdots, D_{M,\mathrm{L}}(t)\}. \quad (4)$$

1.4 Local execution model

At a specific time frame, the wireless channel gain between the MEC server and the ith WD is denoted by h_i. At the beginning of each time frame, WDs harvest energy from the MEC server, and eT is used for WPT. The ith WD harvests energy E_{H} from the MEC server:

$$E_{\mathrm{H}} = \mu P h_i e T, \quad (5)$$

where P denotes the transmit power of the MEC server; $\mu \in (0, 1)$ and represents the energy harvesting efficiency of WDs. When tasks are computed locally, the WDs' processor is the primary source of energy consumption. Therefore, the square of the WD's processor frequency is directly proportional to its energy consumption. In the local computing mode, a specific WD can locally execute its computation tasks and harvest energy at the same time[21-22]. Denote the WD's computation (cycle per second) by f_i. Similarly, the computation time is denoted by t_i which follows constraint $0 \leqslant t_i \leqslant T$[22-23]. The amount of the data processed by the WD is $f_i t_i = \phi$, where $\phi > 0$ and it denotes the number of cycles needed to process one bit of task data.

Each WD must first process the prioritized computing task within a fixed time frame using the harvested energy from the MEC server. The ith WD is given a unique weight w_i, and a higher w_i means a higher computation rate. Meanwhile, $E_{i,\mathrm{L}}$ represents the ith WD's energy consumption of the local computing:

$$E_{i,\mathrm{L}} = k_i f_i^3 t_i, \quad (6)$$

where k_i is energy efficiency coefficient[10]. In general, we assume $E_{\mathrm{H}} \geqslant E_{i,\mathrm{L}}$ to be necessary for WDs to work properly.

It is obvious that the harvested energy of a WD should be completely used, and the WD should compute tasks throughout the time frame T in order to process as many tasks as possible. That is, $t_i^* = T$ and subsequently $f_i^* = \left(\frac{E_{i,\mathrm{L}}}{k_i T}\right)^{\frac{1}{3}}$. Therefore, the rate of local computation (bit per second) is given by

$$r_{i,\mathrm{L}}^*(e) = \frac{f_i^* t_i^*}{\phi T} = \frac{(\mu e P)^{\frac{1}{3}}}{\phi}\left(\frac{h_i}{k_i}\right)^{\frac{1}{3}}. \quad (7)$$

Here assume that it will schedule the WD processor frequency according to the policy of maximizing the computation rate. $D_{i,\mathrm{L}}(t)$ represents the size of the tasks computed locally by the processor of the ith WD.

$$\frac{D_{i,\mathrm{L}}(t)}{|T|} = r_{i,\mathrm{L}}^*(e), i \in M, t \in T, \quad (8)$$

where $|T|$ denotes the length of a computation time frame.

1.5 Edge computing model

When the mode selection parameter x_i is 1, the tasks are completely sent to the MEC server, and then the corresponding computation results are returned to the

WDs. It is worth noting that the data transfer relies on the condition of the wireless channel gain h. WDs can offload the heavy workload to the MEC server, but it can also increase the processing time delay. As a result, the trade-off between energy usage and time delay should be considered.

Assume that in the edge computing mode, the main energy consumption is resulted from offloading tasks. In order to attain the maximum computation capability, a WD in the edge computing mode fully exhausts its harvested energy to offload the tasks[24]. N_0 denotes the receiver noise power, and the communication bandwidth is denoted by B_w. In the edge execution mode, the tasks sent to the MEC server are represented by $D_{i,\mathrm{E}}(t)$:

$$D_{i,\mathrm{E}}(t)=\frac{B_w\tau_i T}{\nu_\mu}\log_2\left(1+\frac{h_i E_{i,\mathrm{E}}}{N_o\tau_i T}\right),\ i\in M, \tag{9}$$

where $E_{i,\mathrm{E}}$ is the energy consumed for transmitting tasks to the MEC server; ν_μ is the communication overhead ratio. Similarly, power and energy in the edge computing mode follow the relationship $P=E_{i,\mathrm{E}}/(\tau_i T)$. We assume that the communication and WPT between the MEC server and the WDs occur within the similar frequency band. The communication and energy transfer operations between the MEC server and the WDs follow the time division multiplexing (TDM) operating order. A TDM operating order avoids mutual interference in the MEC server[25]. Because of TDM, after harvesting energy, a WD operating in the edge computing mode can offload its tasks to the MEC server[26-27]. As a result, the computation rate in the edge computing mode is equal to the data offloading capacity:

$$r^*_{i,\mathrm{E}}(e,\tau)=\frac{B_w\tau_i}{\nu_\mu}\log_2\left(1+\frac{\mu P e h_i^2}{\tau_i N_0}\right). \tag{10}$$

1.6 Problem formulation

Based on the model discussed above, increment in the buffer queue size definitely increases the workload of the MEC system. Therefore, the computation rate of the system decreases. For each time frame, to obtain a better balance between the computation rate and the buffer queue size, we formalize the weighted sum of the buffer queue size and the computation rate,

$$Q(h,\tau,e,x,A_i)=\sum_{i=1}^{M}w_i(1-x_i)\frac{(\mu e P)^{\frac{1}{3}}}{\phi}\left(\frac{h_i}{k_i}\right)^{\frac{1}{3}}+\sum_{i=1}^{M}w_i x_i\left(\frac{B_w\tau_i}{\nu_\mu}\log_2\left(1+\frac{\mu P e h_i^2}{\tau_i N_0}\right)\right)-\sum_{i=1}^{M}B_i(t), \tag{11}$$

where $\tau=\{\tau_i\mid i\in M\}$; $x=\{x_i\mid i\in M\}$; P and N_0 are fixed parameters. It is obvious from the above Eq. (11), that when $x_i=0$, the tasks are locally computed. Otherwise, the tasks are offloaded to the MEC server when $x_i=1$. Obviously, there is a trade-off between the buffer queue size and the computation rate.

Therefore, our study aims to find an optimal solution that achieves a lower buffer queue size and a higher computation rate simultaneously. Furthermore, with the given task A_i in the buffer queue $B_i(t)$ and the given wireless channel gain h, the problem is formulated as the sum of the computation rate of the MEC server executing the offloaded tasks, the local computation rate and negative of the buffer queue size. In this work, we aim to maximize the following problem.

$$\text{Problem 1: } \max_{h,\tau,e,x,A_i} Q(h,\tau,e,x,A_i)$$

s.t. Eq. (1),

$$\tau_i\geqslant 0,\ e\geqslant 0,\ \forall i\in M, \tag{12}$$

$$x_i\in\{0,1\}, \tag{13}$$

$$B_i=x_i A_i,\ x_i\in\{0,1\}. \tag{14}$$

It is inferred that if $x_i=0$, the MEC computation time $\tau=0$. It happens when the ith device chooses the local execution mode. In the local execution mode for the ith device, the buffer queue size will also become zero. Therefore, we have to find the offloading decision x_i before pursuing the optimal solution. Otherwise, the Problem 1 mentioned above is a non-convex and mixed integer optimization programming problem. It is difficult to find a solution for such a non-convex and mixed integer optimization programming problem. Thus, it should be transformed to a convex optimization programming problem. Once the fixed values about the mode selection parameter x are obtained, Problem 1 becomes a convex optimization programming problem.

$$\text{Problem 2: } \max_{h,x,A_i} Q(h,\tau,e,x,A_i)$$

s.t. Eqs. (1), (12) and (14). The observations above motivate us to devise a deep RL-based algorithm to solve such a convex optimization programming problem stated in Problem 2.

2 Deep RL-Based Task Offloading Algorithm

We address Problem 2 by developing a deep RL-based task offloading algorithm. As it is analyzed in the previous section, Problem 1 can be transformed to a convex optimization programming problem easily with the given x. Therefore, the proposed deep RL-based task offloading algorithm follows two steps, which is shown in Fig. 4. The first step determines the appropriate e and τ. The optimal set $\{e,\tau\}$ maximizes the $Q(h,\tau,e,x_i,A_i)$ with given h_i, x and A_i. Thus, the channel gain h_i and the corresponding output of the deep neural network (DNN) x will be used as a training set. The second step is to train the DNN with the obtained training set from step 1. The workflow of the deep RL-based task offloading algorithm is shown in Fig. 4.

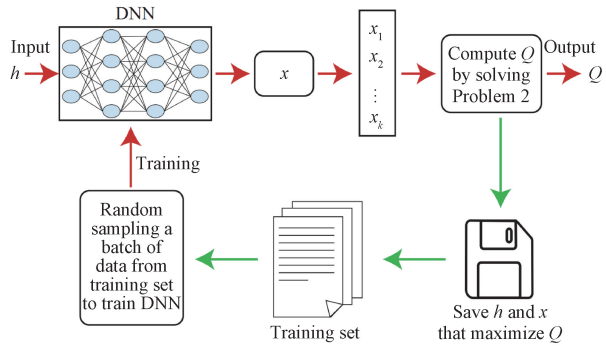

Fig. 4 Workflow of deep RL-based task offloading algorithm

2.1 Optimal time allocation $\{e, \tau\}$

In this part, we deal with optimal time allocation $\{e, \tau\}$ problem. We find a solution $\{e, \tau\}$ for Problem 2 with the given x and A_i. To solve Problem 2, we introduce a Lagrangian multiplier ς to Eq. (11). The converted Lagrangian form of Problem 2 is

$$L(e, \tau, \varsigma) = \sum_{i=1}^{M} w_i(1 - x_i)\eta_1\left(\frac{h_i}{k_i}\right)^{\frac{1}{3}} e^{\frac{1}{3}} + \sum_{i=1}^{M} w_i x_i \varepsilon \tau_i \ln\left(1 + \frac{\eta_2 e h_i^2}{\tau_i}\right) - \sum_{i=1}^{M} x_i A_i(t) + \varsigma\left(1 - e + \sum_{i=1}^{M} \tau_i\right), \quad (15)$$

s. t. $e, \tau \geqslant 0$,

where η_1, η_2 and ε are constant parameters, $\eta_1 = \frac{(\mu P)^{\frac{1}{3}}}{\phi}$, $\eta_2 = \frac{\mu P}{N_0}$ and $\varepsilon = \frac{B_w}{\nu_\mu \ln 2}$.

The corresponding dual function would be

$$d(\varsigma) = \max_{e,\tau}\{L(e, \tau, \varsigma) \mid e \geqslant o, \tau \geqslant 0, \forall i \in M\}. \quad (16)$$

Then, we have a dual problem as

$$\min_{\varsigma}\{d(\varsigma) \mid \varsigma \geqslant 0\}. \quad (17)$$

Once the optimal time $\{e^*, \tau^*\}$ is determined, the dual problem given in Eq. (17) can be easily solved. The optimal time satisfies

$$e + \sum_{i=0}^{M} \tau_i = 1. \quad (18)$$

Theorem 1

There exists one optimal solution $\{e^*, \tau^*, \varsigma^*\}$ that satisfies

$$\frac{\tau^*}{e^*} = \frac{\eta_2 h_i^2}{-1 - W\left(\frac{-1}{\exp\left(1 + \frac{\varsigma^*}{w_i \varepsilon}\right)}\right)}, \quad \forall_i \in M, \quad (19)$$

where $W(y)$ is the Lambert – W function, i. e., $u = W(y)$. Lambert – W function is an inverse function of $f(u) = u \cdot \exp(u) = y$.

Proof

The partial derivative of the parameter L in Eq. (15) with regard to the parameter τ^* is

$$\frac{\partial L}{\partial \tau^*} = w_i \varepsilon \ln\left(1 + \frac{\eta_2 e h_i^2}{\tau_i}\right) - \frac{w_i \varepsilon \eta_2 e h_i^2 (1/\tau^*)^{-1}}{1 + \eta_2 e h_i^2 (1/\tau^*)^{-1}} - \varsigma^*. \quad (20)$$

By setting $\frac{\partial L}{\partial \tau^*} = 0$ at the optimal point, we have

$$\ln\left(1 + \frac{\eta_2 e h_i^2}{\tau^*}\right) + \frac{1}{1 + \eta_2 e h_i^2/\tau^*} = 1 + \frac{\varsigma^*}{w_i \varepsilon}. \quad (21)$$

Taking exponential on both sides, we get

$$\ln\left(1 + \frac{\eta_2 e h_i^2}{\tau^*}\right) \exp\left(\frac{1}{1 + \eta_2 e h_i^2/\tau^*}\right) = \exp\left(1 + \frac{\varsigma^*}{w_i \varepsilon}\right). \quad (22)$$

Denote two positive values a and b that satisfy:

$$-a\exp(-a) = -1/b. \quad (23)$$

So, here $a = -W(-1/b)$. Combining Eqs. (22) and (23), we can get Eq. (19).

In summary, Theorem 1 holds.

When $y \in (-1/e, 0)$, subsequently the Lambert – W function $W(y) \in (-1, 0)$. From Eq. (19), we can infer that ς^* must be higher than zero. Otherwise, either $\tau_i^* \to \infty$ or $e^* \to 0$. Similarly, when $y \in (-1/e, 0)$, the the Lambert – W function has the increasing behavior. Consequently, the denominator of the right-hand side of Eq. (19) is always higher than zero. Therefore, when $\varsigma^* > 0$, it follows $-1/e < \frac{-1}{\exp\left(1 + \frac{\varsigma}{w_i \varepsilon}\right)} < 0$.

By transforming Eq. (19), we can obtain the optimal τ^*:

$$\tau^* = e^* \eta_2 h_i^2 \psi_i(\varsigma^*), \quad (24)$$

$$\psi_i(\varsigma^*) \triangleq \left(-1\left(W\left(\frac{-1}{\exp\left(1 + \frac{\varsigma^*}{w_i \varepsilon}\right)}\right)\right)^{-1}\right)^{-1}. \quad (25)$$

It can be seen that Eq. (18) holds strictly based on the above analyses. Accordingly, by combining Eqs. (18) and (24), as a function of ς^*, we achieve a semi-closed-form e^*:

$$e^* = \frac{1}{1 + \eta_2 \sum_{i \in M} h_i^2 \psi_i \varsigma^*} \triangleq \Pi(\varsigma^*). \quad (26)$$

Both $\psi_i(\varsigma^*)$ and $\Pi(\varsigma^*)$ are functions of the same variable ς^*. The key idea is to find the optimal value for ς^* to solve Problem 2.

Theorem 2

There exists one optimal solution ς^* that satisfies

$$\Gamma(\varsigma^*) \triangleq \frac{1}{3}(\Pi(\varsigma)^*)^{-\frac{2}{3}} \sum_{i\in M} \eta_1 w_i (1-x_i)(h_i/k_i)^{\frac{1}{3}} + \eta_2 \varepsilon \sum_{i\in M} \frac{w_i x_i h_i^2}{1+(\psi(\varsigma^*))^{-1}} - \varsigma^* = 0. \quad (27)$$

From Eq. (27), $\Gamma(\varsigma^*)$ monotonically decreases when $\varsigma^* > 0$.

Proof

We set the $\frac{\partial L}{\partial e^*}$ in Eq. (15) to zero, and we will get ς^* that maximizes L.

$$\frac{\partial L}{\partial e^*} = \frac{1}{3}\eta_1 (e^*)^{-\frac{2}{3}}(1-x_i)\sum_{i\in M} w_i (h_i/k_i)^{\frac{1}{3}} + \varepsilon\eta_2 \sum_{i\in M} \frac{w_i x_i h_i^2}{1+\eta_2 e^* h_i^2/\tau_i^*} - \varsigma^* = 0. \quad (28)$$

From Eq. (24), we get that

$$\frac{\eta_2 e^* h_i^2}{\tau_i^*} = \frac{1}{\psi_i(\varsigma^*)}. \quad (29)$$

$$\Gamma(\varsigma^*) \triangleq \frac{1}{3}(\Pi(\varsigma)^*)^{-\frac{2}{3}} \sum_{i\in M} \eta_1 w_i (1-x_i)(h_i/k_i)^{\frac{1}{3}} + \eta_2 \varepsilon \sum_{i\in M} \frac{w_i x_i h_i^2}{1+(\psi(\varsigma^*))^{-1}} - \varsigma^*. \quad (30)$$

By making $\Gamma(\varsigma^*)$ equal to zero, we can find ς, e and τ.

Therefore, Theorem 2 holds.

2.2 Training set

According to the above analyses, it is evident that the mode selection parameter x plays a vital role in solving Problem 2. We already know that a random x at a time frame $t = 1$ is not a satisfactory solution. In the following, we will design a training set to train the deep RL-based task offloading algorithm that gives us an optimal solution for the mode selection parameter (i. e., offloading decision) for Problem 2.

At the beginning of the time frame $t = 1$, we set parameters of the DNN randomly, following a zero-mean distribution. The bias of the DNN is initialized to 0.1. At the same time, we set the length of the arrived tasks randomly in the range of [0.5, 1.5] cycle/bit. Thus, we need to train the DNN to achieve the optimal value for x as an output, given the channel gain h as the input to the DNN.

2.2.1 Finding a better offloading decision x

When a task arrives at the task buffer queue, it might be sent to the MEC server or locally executed. The activation function in the DNN output layer is sigmoid:

$$S(y) = \frac{1}{1+e^{-y}}. \quad (31)$$

Consequently, based on the output of DNN $\bar{x} \in (0, 1)$, we can create K distinct alternative optimal $\bar{\boldsymbol{x}}_K$. Then, we can select the optimal solution among these different $\bar{\boldsymbol{x}}_K$. In the proposed algorithm, K meets the constraint $K \leqslant M + 1$. The first optimal offloading decision $\bar{x}_{1,m}$ can be generated through $\bar{x}$:

$$\bar{x}_{i,m} = \begin{cases} 1, & \text{if } \bar{x}_m > 0.5; \\ 0, & \text{if } \bar{x}_m \leqslant 0.5; \end{cases} \quad \text{for } m = 1, 2, \cdots, M. \quad (32)$$

Then, based on the first offloading decision, we can generate the remaining $\bar{x}_K$, which is formalized as

$$\bar{x}_{i,m} = \begin{cases} 1, & \text{if } \bar{x}_m > \bar{x}_{m,k-1}; \\ 1, & \text{if } \bar{x}_m = \bar{x}_{m,k-1} \text{ and } \bar{x}_{m,k-1} \leqslant 0.5; \\ 0, & \text{if } \bar{x}_m = \bar{x}_{m,k-1} \text{ and } \bar{x}_{m,k-1} > 0.5; \\ 0, & \text{if } \bar{x}_m \leqslant \bar{x}_{m,k-1}; \end{cases} \quad \text{for } k = 2, 3, \cdots, K;\ m = 1, 2, \cdots, M. \quad (33)$$

For an example, when $\bar{\boldsymbol{x}}_K = [0.4 \quad 0.7 \quad 0.2 \quad 0.9]$ and $K = 4$, accordingly, four mode selection actions generated from $\bar{\boldsymbol{x}}_K$ would be $\bar{\boldsymbol{x}}_1 = [0 \quad 0 \quad 1 \quad 1]$, $\bar{\boldsymbol{x}}_2 = [0 \quad 1 \quad 1 \quad 1]$, $\bar{\boldsymbol{x}}_3 = [0 \quad 0 \quad 0 \quad 1]$ and $\bar{\boldsymbol{x}}_4 = [1 \quad 1 \quad 1 \quad 1]$. In this quantification, the distance between $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}_K$ is larger than pre-existing exhaustive method that generates 2^M actions at most. So, the complexity of the proposed algorithm has been reduced, and it can maintain the similar performance at the same time. Thus, we can find the best action more quickly.

2.2.2 Training set creation

Problem 2 can be solved through the candidate $\bar{x}_{k,1}$. Then, we can find (h, x^*, A_i) that maximizes $Q(h, \tau, e, x^*, A_i)$ and save these optimal values in memory or use them for training DNN in later steps.

2.3 Training DNN

In our system, the DNN is utilized for learning the distribution of x with the given h. Once the number of data in the training set is higher than the batch size ξ, the DNN gets trained immediately. The DNN is trained by using a batch of training data to enhance the efficiency of learning. We use a memory size of 1 024 bits. When the training set has used the memory size ultimately, the newly generated training data starts to replace the previously generated training data. So, the DNN gets learning from the latest training data only which are more beneficial than the old ones.

There is no substantial difference between adjacent time slots because the algorithm only adds or updates one training dataset at one single time slot. It is a useless and inefficient way for training the DNN at each time slot. So, the training process uses data in batches for learning purposes, and it is dependent on the batch size ξ. Algorithm 1 shows the pseudo-code of the deep RL-based task offloading algorithm.

Algorithm 1: Deep RL-based task offloading algorithm

Input: Wireless channel gain h_i at each time frame t.

Initialize DNN variables with default values.
for $t = 1, 2, \cdots, n$ **do**
  Select the proper value of K based on t.
  Choose the right batch size b and training interval ξ.
  Input channel gain h to DNN to get $\bar{x}$.
  Generate spare x_k.
  Calculate Q for all x_k by solving Problem 2.
  Choose $\bar{x}_k$ that maximizes Q.
  Update training set with $\{h, \bar{x}_k\}$.
  if $t \geqslant b$ and the remainder $(t/\xi) = 0$ **then**
    randomly select b training data;
    train the DNN with selected batches of training data.
  end if
end for

Return: An optimal solution $\{x, \tau, e, B_i(t)\}$ to Problem 2.

3 Performance Evaluation

Deep RL-based task offloading algorithm is evaluated. A DNN with four layers was built using the python library TensorFlow.

3.1 Experimental setup

3.1.1 Metrics

The following metrics are used to investigate the proposed algorithm's performance. Specifically, to analyze the convergence performance of the deep RL-based task offloading algorithm, the metrics are used to compute the normalized computation rate Q and the training loss. Specifically, three metrics are utilized for training purposes: the offloading decision parameter, the buffer queue size and the computation rate. Furthermore, Baseline 1 and Baseline 2 are used as benchmarks to compare the efforts of this dissertation.

In Baseline 1, all WDs locally compute their arrived tasks at each time frame. Likewise, in Baseline 2, all WDs offload their arrived tasks to the MEC server at each time frame.

3.1.2 Parameter settings

This research work considers 10 WDs communicating with the MEC server simultaneously based on the proposed algorithm. The computation tasks generated by these WDs either are computed locally by themselves or are offloaded to the MEC server. There are 90 000-time frames in the simulation system.

We make use of the Power cast TX91501-3W as a transmitter at the MEC server with the power P of 3 W. Similarly, we use the P2110 Power Harvester in each WD, and its energy harvesting efficiency μ is 0.51. We set ξ = 10 and M = 10 in default settings. The free space path loss model is considered for wireless channel gain h.

$$\bar{h}_{\mathrm{m}} = A_d \left(\frac{3 \times 10^8}{4\pi f_{\mathrm{c}} d_i} \right)^{d_{\mathrm{e}}}, \tag{34}$$

where $\bar{h}_{\mathrm{m}}$ represents the average channel gain; d_{e} is the path loss exponent, and d_{e} = 2.8 Hz; A_d is the antenna gain; f_{c} is the carrier frequency, and f_{c} = 915 MHz without loss of generality; d_i is the distance between the transmitter and the receiver. We define one-time frame T = 1. A model, Rayleigh fading channel $h_{\mathrm{m}} = \bar{h}_{\mathrm{m}} \beta_{\mathrm{m}}$, is used to generate the wireless channel gain of each WD at each time slot. β_{m} is the independent random channel fading factor and follows a unit mean exponential distribution. Arrived tasks A_i in the buffer queue are measured in the same units as the computation rate. A_i can be any random value in a range of $[0.5, 1.5]$, depending on the task generated from the WDs at a specific time frame. The default parameter settings for the simulation model are given in Table 1.

Table 1 Default settings of parameters

Parameter	Value	Parameter	Value
M	10	ξ	40
μ	0.7	B_{w}/Hz	2×10^6
P/W	3	A_{d}	4.11
d_{e}/Hz	2.8	N_0/W	10^{-10}
f_{c}/MHz	915	ϕ/(cycle/bit)	110

3.2 Simulation results

3.2.1 Convergence

The convergence of the deep RL-based task offloading algorithm in terms of the normalized computation rate and the training loss is given in Figs. 5 and 6, respectively.

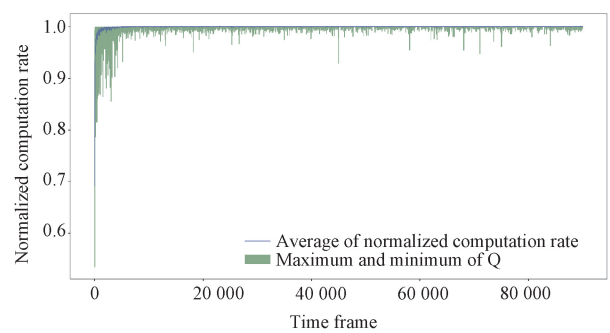

Fig. 5 Normalized computation rate Q versus time frame t

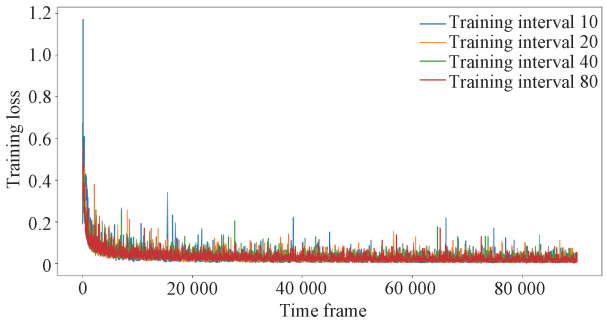

Fig. 6 Change of training loss with training interval ξ

The normalized computation rate $Q(h, x)$ is

$$Q(h, x) = \frac{Q^*(h, x)}{\max\limits_{x^* \in \{0, 1\}^M} Q(h, x^*)}. \tag{35}$$

Its value can occur in the range of $0 \leqslant Q(h, x) \leqslant 1$.

In Fig. 5, the Q-t curve is given for $K = M$ and $M = 10$ as the default parameters. We can see that Q converges to the optimal solution when t increases gradually. Specifically, Q exceeds 0.98 when $t > 400$ and the variance of Q gradually approaches to zero when $t > 2000$.

DNN is developed in the python library called TensorFlow. The DNN training loss is already investigated by adjusting the learning interval ξ of DNN. So, a specific learning interval ξ is considered to study the relationship between the training loss and the algorithm efficiency. The variations in the training loss have been compared at different learning intervals, i.e., 10, 20, 40 and 80. In Fig. 6, the training loss slows down gradually as the learning interval increases.

It can be observed that, when $\xi = 10$, the training loss decreases to 0 quickly, but the operation efficiency of deep RL-based task offloading algorithm remains at a low standard. On the contrary, when $\xi = 80$, the training loss decreases slowly and takes long time before approaching to 0. The operation efficiency of the proposed algorithm has significantly improved. Selecting a moderate value for the learning interval ξ, e.g., 40, proves to be a practical and innovative choice. Furthermore, the training loss declines over time and settles down at near 0.04, which is attributed to the random sampling of the training data as well as the randomly arrived task A_i in the task buffer queue generated by the WDs.

Overall, DNN learns iteratively from the previous best action pairs (h, x) and achieves the improved offloading decision that meets the constraints of the memory space. As old memory is replaced by the new one, DNN only uses the most up-to-dated dataset generated by the updated mode selection parameters for training. This closed loop RL algorithm helps to improve itself as the time progresses, and approaches to convergence at the end.

3.2.2 Offloading decision

The impact of the required processing cycles per bit ϕ of the task data on the MEC system offloading decision and the optimal offloading decision of individual WD is given in Figs. 7 and 8.

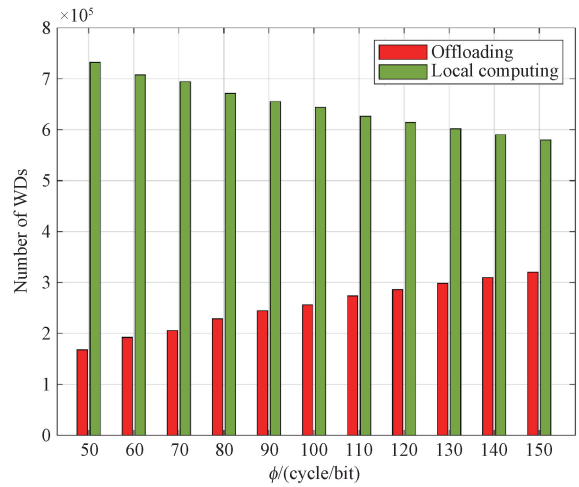

Fig. 7 Offloading decision varying with ϕ

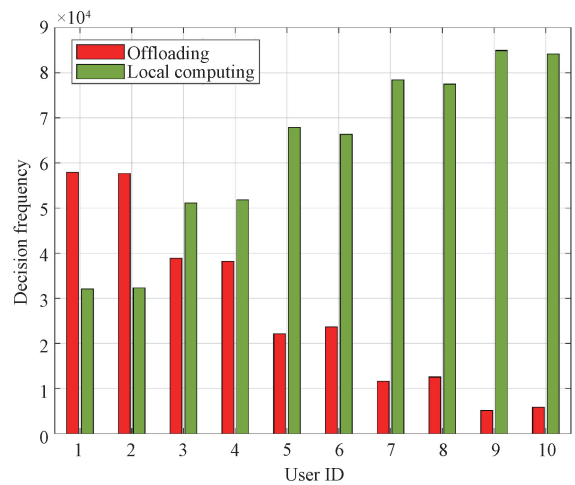

Fig. 8 Each user mode selection frequency

Individually, Fig. 7 shows that offloading decisions varying with ϕ. It can be observed that as ϕ increases, less WDs compute the generated tasks locally, and more WDs prefer to offload their tasks to the MEC server. Specifically, when ϕ is higher than 130, WDs' decisions do not change significantly. One reason is that the WDs' local computation resources are not enough and WDs prefer to offload tasks to the MEC server when ϕ is high.

In addition, we can observe in Fig. 8 that over 90 000 time frames, the number of WDs locally computing the tasks (i.e., the mode selection parameter $x = 0$) has changed a lot with the varying weights and the buffer queue size. The reason is that WDs prefer to offload the tasks to the MEC server at first. Conversely, the increasing the buffer queue size makes WDs choose to work in the local computing mode rather than offloading

tasks to the MEC server. Consequently, our work can make a better trade-off that both the buffer queue size and the time delay (i. e., the computation rate) for processing the generated tasks of WDs remain as low as possible. Thus, it is beneficial to maximize the utilization of computation resources. It has been proved that the proposed deep RL-based task offloading algorithm can adjust the WDs' offloading decisions to maximize Q, with respect to WDs' varying demands of computation resources.

3.2.3 Computation rate

In this section, we investigate the impact of the required cycles ϕ for processing one bit of task data. We make the comparison of the proposed algorithm with two baselines, i. e., Baseline 1 and Baseline 2. As depicted in Fig. 9, the normalized weighted sum of the computation rate in these three algorithms decreases with the increasing ϕ. Moreover, it can be observed that Q in the proposed algorithm is more robust to the increasing ϕ. In addition, Q in the proposed algorithm is higher than that in the other two baselines, which means that the proposed algorithm has better performance in joint optimization of the buffer queue size and the computation rate. It is worth noting that the maximum Q in $Q^* = \frac{Q}{\max Q}$ is obtained when ϕ equals 110.

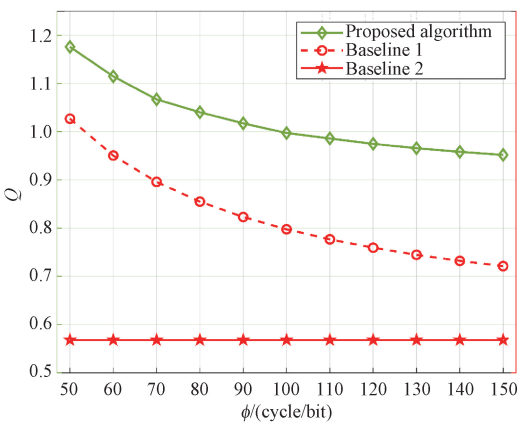

Fig. 9 Impact of ϕ on Q

In addition, it can be observed that Q in Baseline 1 decreases continuously, and Q in Baseline 2 is not affected by ϕ at all. The reasons are that Baseline 1 is only related to the local execution and that Baseline 2 is not constrained by the local computing power.

3.2.4 Buffer queue size

This section investigates the impact of the optimal solution to Problem 2 on the buffer queue size. The impact of the needed cycles for processing one-bit data ϕ and its effect on the number of WDs M with respect to the buffer queue size are given in Figs. 10 and 11, respectively.

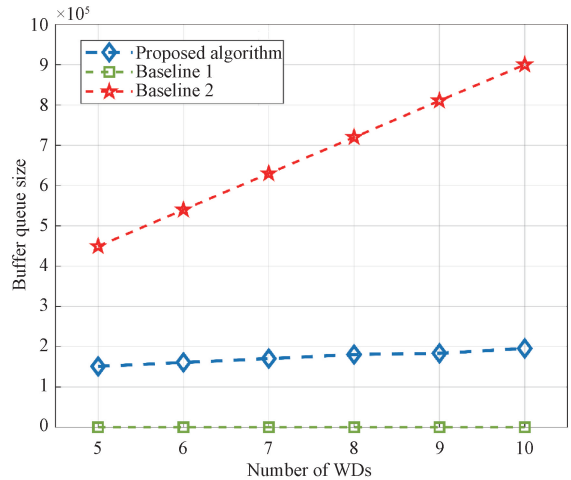

Fig. 10 $B_i(t)$ versus M

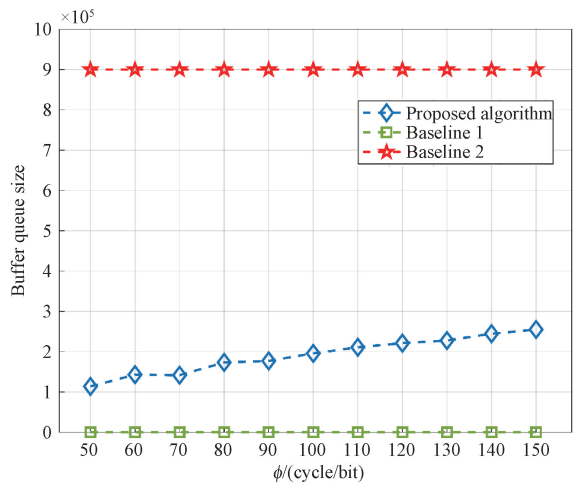

Fig. 11 $B_i(t)$ versus ϕ when $M = 10$

In Fig. 10, the buffer queue size increases as the number of WDs increases under offloading as shown in Baseline 2 and the proposed algorithm. Subsequently, the size of generated tasks also increases with the increase in the number of WDs. It is obvious that the increase in the buffer queue size with the increasing number of WDs is not as significant as it occurs with a single device. The reason is that the buffer queue deals with the arrived tasks from WDs and then the proposed algorithm makes sure that the buffer queue size remains at minimum. So, it prefers the local execution rather than the MEC execution. As a result, tasks remain in the buffer queue are those only offloaded to the MEC server. In addition, it is obvious that there are no tasks in the buffer queue when $x = 0$. Similarly, in Baseline 2, when all WDs choose to send their tasks to the MEC server, the buffer queue size is maximum for each WD.

Similarly, Fig. 11 shows the impact of ϕ on the buffer queue size. As it is analyzed above, as the cycles required for processing a single bit of task data increases, the number of WDs offloading their tasks also increases,

which subsequently makes increment in the buffer queue size $B_i(t)$.

The reason is that as ϕ increases, the energy consumption of processing their tasks in local execution increases. So, more WDs send their tasks to the MEC server, which in turn makes increment in the buffer queue size. In Baseline 2, the buffer queue size is not affected by ϕ.

3.2.5 Discussion

In this section, we compare the training loss of the proposed algorithm with the selfish-aware task offloading algorithm[28] along individual time frames as shown in Fig. 12. The selfish-aware task offloading method is selfish aware of the users in the MEC network trying to get more computation resources so it punishes them to give more computation resources to the normal users in the MEC network. In the beginning, the training loss curve shows the minimum loss as the algorithm initializes to recognize the selfish users. Once the proposed algorithm recognizes the selfish users, it starts optimizing computation resources accordingly. Meanwhile, the training loss of the proposed algorithm follows the conventional declining curve with time, reducing the loss to the minimum. The proposed algorithm is focused on enhancing the computation resources of the WDs and minimizes the task buffer queue size in WDs.

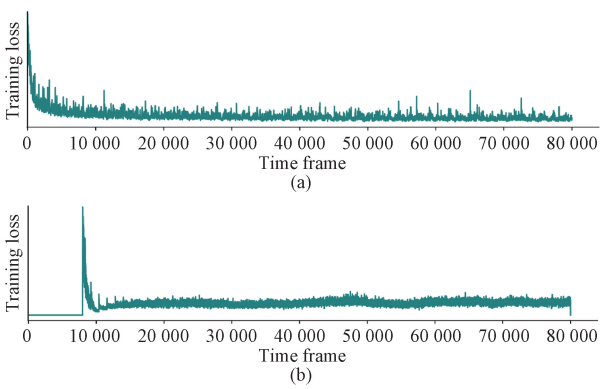

Fig. 12 Training loss comparison between proposed algorithm and selfish-aware task offloading algorithm over time: (a) proposed algorithm; (b) selfish-aware task offloading algorithm

4 Conclusions

This research work proposes a deep RL-based task offloading algorithm for a wireless-powered and multiple-user MEC system with the buffer queue at the terminal devices. The proposed algorithm maximizes the weighted sum computation rate and minimizes the buffer queue size for user devices connected to the MEC system. A mechanism for adaptive parameter setting is developed to achieve rapid algorithm convergence. The proposed algorithm utilizes RL to enhance its offloading actions prescribed by a DNN by learning from previous offloading experiences. Conventional optimization approaches require tackling complex mixed integer optimization programming problems. However, the proposed algorithm eliminates this requirement by converting it to a convex optimization programming problem. The simulation findings reveal that the proposed algorithm achieves near-optimal performance while reducing CPU execution delay by more than an order of magnitude, making real-time system optimization realistic for wireless-powered MEC networks in fading environments. In summary, extensive simulation results demonstrate that the proposed algorithm can maintain an elevated level of the computation rate while making the buffer queue size at a minimum.

The study is highly significant because deep RL-based algorithms are taking the place of conventional optimization techniques and finding their usage in different real-life scenarios. For the MEC system, this work explores a better use of the deep RL-based algorithm that leads industrial IoT devices to integrate into the new paradigm of the MEC system. This integration solves a pervasive issue of MEC in a robust manner, and meanwhile proves its best performance.

References

[1] VAILSHERY L S. Iot and non-iot connections worldwide 2010–2025[EB/OL]. (2022-09-22)[2023-10-12]. https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide.

[2] HU Y C, PATEL M, SABELLA D, et al. Mobile edge computing: a key technology towards 5G [J]. *ETSI White Paper*, 2015, 11(11): 1-16.

[3] XIAO Z, DAI X X, JIANG H B, et al. Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method [J]. *IEEE Internet of Things Journal*, 2020, 7(3): 2038-2052.

[4] XIAO Z, XIAO D P, HAVYARIMANA V, et al. Toward accurate vehicle state estimation under non-gaussian noises[J]. *IEEE Internet of Things Journal*, 2019, 6(6): 10652-10664.

[5] MACH P, BECVAR Z. Mobile edge computing: a survey on architecture and computation offloading[J]. *IEEE Communications Surveys & Tutorials*, 2017, 19(3): 1628-1656.

[6] ABBAS N, ZHANG Y, TAHERKORDI A, et al. Mobile edge computing: a survey [J]. *IEEE Internet of Things Journal*, 2018, 5(1): 450-465.

[7] PSIHA M M, VLAMOS P. IoT applications with 5G connectivity in medical tourism sector management: third-party service scenarios[M]// VLAMOS P. Advances in Experimental Medicine and Biology. Cham: Springer International Publishing, 2017: 141-154.

[8] LIU Y Q, PENG M G, SHOU G C, et al. Toward edge intelligence: multiaccess edge computing for 5G and Internet of Things [J]. *IEEE Internet of Things Journal*, 2020, 7(8): 6722-6747.

[9] XIAO Z, LI F C, JIANG H B, et al. A joint information and energy cooperation framework for CR-enabled macro-femto heterogeneous networks [J]. *IEEE Internet of Things Journal*, 2020, 7(4): 2828-2839.

[10] ZHANG G L, NI S F, ZHAO P. Learning-based joint optimization of energy delay and privacy in multiple-user edge-cloud collaboration MEC systems [J]. *IEEE Internet of Things Journal*, 2022, 9(2): 1491-1502.

[11] ZHAO P, TAO J W, RAUF A, et al. Load balancing for energy-harvesting mobile edge computing [J]. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2021, E104. A(1): 336-342.

[12] ZHANG G L, ZHANG W Q, CAO Y, et al. Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices [J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(10): 4642-4655.

[13] LIN Z F, WANG F, LIU L C. Computation rate maximization for multiuser mobile edge computing systems with dynamic energy arrivals [C]//2021 IEEE/CIC International Conference on Communications in China (ICCC). New York: IEEE, 2021: 312-317.

[14] ABDUL R, MUHAMMAD U, AHMAD B, et al. Health monitoring of induction motor using electrical signature analysis [J]. *Journal of Donghua University (English Edition)*, 2022, 39(3): 265-271.

[15] XU L T, HUANG D Y, ALI ZAIDI S F, et al. Graph Fourier transform based audio zero-watermarking [J]. *IEEE Signal Processing Letters*, 2021, 28: 1943-1947.

[16] YANG Y, HU Y L, GURSOY M C. Deep reinforcement learning and optimization based green mobile edge computing [C]//2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC). New York: IEEE, 2021: 1-2.

[17] ZHAN W H, LUO C B, WANG J, et al. Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing [J]. *IEEE Internet of Things Journal*, 2020, 7(6): 5449-5465.

[18] YU S, CHEN X, ZHOU Z, et al. When deep reinforcement learning meets federated learning: intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network [J]. *IEEE Internet of Things Journal*, 2021, 8(4): 2238-2251.

[19] BI S Z, ZHANG Y J. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading [J]. *IEEE Transactions on Wireless Communications*, 2018, 17(6): 4177-4190.

[20] LI J, ZENG F Z, XIAO Z, et al. Drive2friends: inferring social relationships from individual vehicle mobility data [J]. *IEEE Internet of Things Journal*, 2020, 7(6): 5116-5127.

[21] WANG F, XU J, WANG X, et al. Joint offloading and computing optimization in wireless powered mobile-edge computing systems [C]// 2017 IEEE International Conference on Communications (ICC). New York: IEEE, 2017: 1784-1797.

[22] JIANG H B, LI J, ZHAO P, et al. Location privacy-preserving mechanisms in location-based services [J]. *ACM Computing Surveys*, 2022, 54(1): 1-36.

[23] HERBERT S, WASSELL I, LOH T H, et al. Characterizing the spectral properties and time variation of the in-vehicle wireless communication channel [J]. *IEEE Transactions on Communications*, 2014, 62(7): 2390-2399.

[24] WU X D, FU L Y, WANG S Q, et al. Collective influence maximization in mobile social networks [J]. *IEEE Transactions on Mobile Computing*, 2023, 22(2): 797-812.

[25] LIU D B, CAO Z C, HOU M S, et al. Pushing the limits of transmission concurrency for low power wireless networks [J]. *ACM Transactions on Sensor Networks*, 2020, 16(4): 1-29.

[26] PRABHA B, RAMESH K, RENJITH P N. A review on dynamic virtual machine consolidation approaches for energy-efficient cloud data centers [M]//Data Intelligence and Cognitive Informatics. Singapore: Springer Singapore, 2021: 761-780.

[27] JIANG H B, LIU W P, JIANG G Y, et al. Fly-navi: a novel indoor navigation system with on-the-fly map generation [J]. *IEEE Transactions on Mobile Computing*, 2021, 20(9): 2820-2834.

[28] ZHAO P, YANG Z Y, MU Y Q, et al. Selfish-aware and learning-aided computation offloading for edge-cloud collaboration network [J]. *IEEE Internet of Things Journal*, 2023, 10(11): 9953-9965.

具有缓存队列的无线供电和多用户移动边缘计算系统的计算速率最大化

ABDUL Rauf*，赵　萍
东华大学 信息科学与技术学院，上海 201620

摘　要：移动边缘计算（mobile edge computing，MEC）在各种延迟敏感型应用中发挥着至关重要的作用。随着工业 4.0 技术中低计算能力物联网（Internet of Things，IoT）设备的日益普及，MEC 可促进无线电传输，提高设备效率和可持续性。与 MEC 计算速率最大化相关的研究包括坐标下降法、基于交替方向乘子法（alternating direction method of multiplier，ADMM）和 Lyapunov 优化。然而，这些研究没有考虑缓冲队列的大小。该文关注具有缓冲队列的无线供电和多用户 MEC 系统的计算速率最大化，并提出了一种基于深度强化学习（reinforcement learning，RL）的任务卸载算法。该算法最大化计算速率并最小化缓冲队列的大小。其主要思路是探索连接到 MEC 系统的物联网设备的最佳模式选择。该文提出基于特定时隙中的单个通道增益、缓冲队列大小和无线电力传输最大化的模式选择。在此基础上，进一步将任务卸载问题形式化为缓冲队列大小和计算速率。然后，设计了一种基于深度 RL 的卸载算法来解决这种混合整数非凸优化问题，旨在更好地权衡缓冲队列大小和计算速率。大量的仿真结果表明，所提算法比现有算法更有效，能同时保持较小的缓冲队列和较大的计算速率。

关键词：移动边缘计算（MEC）；计算率；缓冲队列；非凸优化；深度强化学习