Received: 24 November 2021 | Accepted: 22 January 2022

DOI: 10.1002/int.22841

RESEARCH ARTICLE

WILEY

Deep reinforcement learning-based multitask hybrid computing offloading for multiaccess edge computing

Jun Cai | **Hongtian Fu** | **Yan Liu**

School of Cyber Security, Guangdong Polytechnic Normal University, Guangzhou, Guangzhong, China

Correspondence
Yan Liu, School of Cyber Security, Guangdong Polytechnic Normal University, 510665 Guangzhou, Guangzhong, China.
Email: liuyan_sysu@163.com

Abstract

By deploying computing units in edge servers, the device-generated computation-intensive tasks can be offloaded from the cloud, lessening the core network's traffic and reducing the tasks' completion latency. To mitigate the burden on edge server and improve user experience, this paper proposes a deep reinforcement learning (DRL)-based multiuser multitask hybrid computing offloading model for offloading a set of computation-intensive tasks generated by multiple users to edge server and adjacent devices. The proposed model makes global computing offloading decisions for multiple computation-intensive tasks simultaneously rather than via one-by-one decision-making, which takes the impact of users' offloading decisions on the system's overall performance in multitask offloading scenarios into account. The main goal of this study is to reduce the long-term overall system delay. The model uses the recurrent neural network to extract the feature information of task and network state, improving the convergence speed and stability of the DRL model. The experimental results demonstrate that the global offloading decision-making model outperforms other methods regarding long-term overall system delay and device energy consumption.

KEYWORDS

deep reinforcement learning, edge computing, hybrid computing offloading, multitask, recurrent neural network

1 INTRODUCTION

The rapid development of artificial intelligence and wireless communication technology enabled the emergence of many computation-intensive applications, such as virtual reality, image recognition, and automatic driving. These applications are sensitive to delay and require extensive computational resources.[1,2] The software and hardware technologies of smartphones, computers, and other smart devices are developing rapidly, and the Central Processing Units (CPUs) and Graphics Processing Units (GPUs) in these devices are becoming more and more powerful. However, as the number of applications and the complexity of tasks constantly increase, it is becoming difficult for smart devices to process a large number of applications in a short time. Consequently, it takes longer to accomplish a task in the multitask scenario, potentially leading to task loss when the delay exceeds the time limit.[3–5]

The computing offloading technology appeared as a response to the need to reduce the computational delay and improve the service quality. Computing offloading refers to transferring local tasks to other resource-rich network nodes for more efficient data processing. Offloading the tasks generated by devices to the resource-rich cloud resolves the contradiction between the large amount of computing resources required by computing-intensive tasks and the limited computing resources available on each terminal device.[6] However, although cloud computing technology provides users with far more computing, storage, and other resources than locally available, it suffers several drawbacks. Namely, cloud computing centers are usually deployed far away from users and cannot respond to delay-sensitive tasks in real-time. In addition, the transmission of massive data brings great pressure to the backhaul link and backbone network, also risking privacy disclosure in the communication process.

To tackle the discussed shortcomings of cloud computing, the European Telecommunications Standardization Association proposed the multiaccess edge computing (MEC) technology.[7] MEC is envisioned as a supplement to cloud computing. By deploying computing, storage, and other resources near the devices, MEC enables swift response to user service requests and reduces network traffic and cloud computing load.[8–10] Cloud computing is implemented in a fully centralized manner, with servers typically concentrated in one or several locations. In contrast, edge computing is implemented at the network's edge in a fully distributed manner. Edge computing enables reducing the completion latency and device energy consumption, consequently improving user experience by offloading the tasks generated by devices to the edge computing servers close to the users for execution.[11–13] Therefore, edge computing can significantly reduce delay and jitter compared with cloud computing.

However, while edge computing has many advantages over traditional network architecture, only a limited quantity of computing and storage resources can be provided in this computing environment. With the explosive growth in the number of devices, the number of tasks that need to be offloaded to edge computing servers increases, bringing a huge burden to the resource allocation and uplink utilization of the edge computing server.[14–16] Research reveals that the CPU utilization of a device is commonly less than one-third.[17] Resource-deficient devices offload tasks to adjacent resource-rich idle devices through device-to-device (D2D) communication, thus effectively reducing the server's computing load and uplink traffic.[18] The fusion of D2D and MEC can enhance the computing offloading capability, further improving the system's computing power. Therefore, device-edge hybrid computing offloading is an important research direction in the computing offloading domain.

Scholars have intensively studied computing offloading in the edge computing environment. However, existing studies often do not consider the impact of users' offloading decisions on the system's overall performance in multitask computing offloading scenarios. In most real-world applications, multiple devices are interconnected and cooperate with each other. As a result, the different devices' tasks affect each other and jointly determine the system performance. For example, when dealing with the problem of offloading the data generated by multiple smart cameras in an industrial video processing system, one needs to optimize the overall system delay rather than the delay of a single video processing task.[19] In a time-varying environment with random arrival of tasks, selecting appropriate offloading nodes according to users' computing requirements to improve the system's overall service quality represents one of the key problems of device-edge hybrid computing offloading. Thus, this paper proposes a deep reinforcement learning-based multitask hybrid computing offloading (DRL-MHO) model for the device-edge hybrid computing offloading scenario involving multiple access connections and multiple tasks. The proposed model makes global offloading decisions for multiple tasks simultaneously. It offloads the tasks to the edge computing server or adjacent devices according to the task state and available system resources, thus reducing the long-term delay of the overall system.

The main contributions of this study are as summarized as follows.

- A deep reinforcement learning (DRL)-based multitask device-edge hybrid computing offloading model is proposed for making global offloading decisions for multiple tasks simultaneously, which takes the interaction between multiple tasks into consideration. It aims to minimize long-term system delay instead of completion time of individual tasks.
- To accelerate the convergence and improve stability of the DRL model with high-dimensional state and action space, a recurrent neural network (RNN) model is introduced to extract the feature information of task status and network state.
- Experimental results show that the proposed model can reduce long-term system delay and device energy consumption compared with other offloading algorithms. Further, compared with conventional DRL models, the DRL model based on the RNN has faster convergence speed and higher stability.

The remainder of this paper is arranged as follows. Section 2 introduces existing work. Section 3 describes the system model, whereas Section 4 details the offloading model proposed in this paper. Section 5 describes the experimental environment and analyzes the experimental results. Finally, Section 6 concludes this study with an outlook of future research.

2 | RELATED WORK

Task offloading algorithm is a core component of edge computing architecture, directly affecting network performance and user experience. Several solutions to the task offloading problem in the MEC scenarios have been proposed. According to the offloading mode, the existing computing offloading methods can be roughly classified as device-to-edge (D2E) offloading and D2D–D2E hybrid offloading.

2.1 | D2E offloading

For D2E offloading, the task can be either executed locally, or offloaded to MEC server. Huang et al.[20] proposed a DRL-based online offloading framework to make decisions for task offloading and wireless resource allocation. Lei et al.[21] proposed a joint task offloading and multiuser scheduling algorithm to minimize the weighted sum of computational latency and energy consumption in Internet of Things (IoT) edge computing. Li et al.[22] proposed a task offloading and resource allocation method based on reinforcement learning to reduce task completion latency and device energy consumption. Zhao et al.[23] proposed a DRL-based task offloading algorithm for vehicle edge network. The algorithm prioritizes different tasks, determines the weight coefficients of the tasks, and uses the deep Q network to select the best offloading nodes for tasks, effectively improving the vehicle terminal's task execution efficiency. Guo et al.[24] studied computing offloading in the multitask MEC scenario involving dynamic communication and computing and designed a computing offloading algorithm under the constraint of energy consumption. The algorithm makes decision on computing offloading and energy allocation to optimize completion latency of individual tasks. Liu et al.[25] proposed a distributed offloading algorithm based on multiagent reinforcement learning to minimize the average task completion latency in multiuser MEC scenarios while considering user mobility. Wu et al.[26] proposed a multitask computing offloading scheme for the edge computing scenario involving multiple MEC servers. The scheme offloads the tasks to appropriate MEC servers for processing to minimize the total energy consumption of devices. Sahni et al.[27] proposed a heuristic algorithm that combines partial computing offloading and traffic scheduling for the MEC scenario. The algorithm aims to minimize the average completion time of all tasks. Chen et al.[28] proposed a multiagent DRL-based offloading algorithm to minimize the long-term average network cost under the constraints of latency and energy consumption.

2.2 | D2D–D2E hybrid offloading

For D2D–D2E hybrid offloading, the task can be executed locally, offloaded to adjacent device or MEC server. Feng et al.[29] proposed an online D2D–D2E computing offloading method based on the Lyapunov optimization algorithm to reduce the cost of communication and computation. Seng et al.[30] used the improved Gale–Shapley matching algorithm to find the best matching between tasks and edge servers and devices. The authors also proposed a blockchain-based distributed computing offloading method to reduce servers' computing load and improve the performance of device task processing. Yu et al.[31] proposed a social-aware hybrid computing offloading algorithm for MEC scenarios. The algorithm divides the task into two parts: non-offloadable and offloadable. The offloadable part can be offloaded to device or MEC server. Peng et al.[32] designed a general framework consisting of an application partition, cooperative transmission scheduling, and computation allocating to jointly minimize latency and energy consumption. To maximize the number of devices supported by the cellular networks, He et al.[33] decomposed the D2D–D2E offloading problem into two subproblems, one for minimizing the required edge computation resource for a given D2D pair, one for maximizing the number of devices supported by the cellular network. Li et al.[34] proposed a DRL-based D2D–D2E offloading algorithm to minimizing long-term energy consumption, which takes user mobility into consideration. Jiang et al.[35] proposed a multipoint collaborative computing offloading algorithm based on Lyapunov optimization to jointly optimize the system utility, device energy consumption, and task execution delay. To minimize the total system energy consumption of cognitive wireless networks, Cheng et al.[36] proposed a D2D-aided offloading strategy for nonorthogonal multiple access-enabled cognitive radio networks. The block coordinate descent method and successive convex approximation are used for making decisions for offloading and power control.

Most existing studies offload multiple tasks via one-by-one decision-making and do not consider the impact of decision for one task to other tasks. When multiple tasks need to be offloaded simultaneously, ignoring the resource competition between user devices often leads to improper resource allocation, thus affecting the system's overall performance. Therefore, this paper proposes a DRL-based multitask device-edge hybrid computing offloading model, that is, the D2D–D2E hybrid offloading model for the edge computing scenarios involving multiple devices and tasks. This model considers the interaction between multiple tasks and makes global offloading decisions simultaneously for multiple tasks based on the state of tasks and network resources, to reduce the long-term system computational delay and improve resource utilization and user service quality.

3 | SYSTEM MODEL

In this section, a system model for multiaccess multitask device-edge hybrid computing offloading scenario is introduced, including network, communication, and computing models. Then the global offloading problem is formulated as an optimization problem for minimizing the total long-term system latency.

3.1 | Network model

This paper discusses the quasistatic scenario involving a single MEC-enhanced base station (eNB) and multiple access connections (Figure 1). The time is divided into discrete time slots, denoted as $\mathcal{T} = \{1, 2, ..., T\}$. In each time slot, the device positions remain unchanged, and the wireless channels remain stable.[37] Let $\mathcal{N} = \{1, 2, ..., N\}$ represent the set of devices within the base station's coverage. User devices that generate tasks are called task devices (TDs) and are denoted by $\mathcal{M} = \{1, 2, ..., M\}$. Idle devices with abundant computing resources are called service devices (SDs) and are denoted by $\mathcal{K} = \{1, 2, ..., K\}$. A device can be a TD or an SD, but not both in the same time slot. A device communicates with the base station through the D2E link (i.e., cellular link) and with other devices within the communication range through the D2D communication link.

This model assumes that the tasks generated by TDs are indivisible and can be executed locally or offloaded to either the edge computing server or SDs for execution. In one time slot,

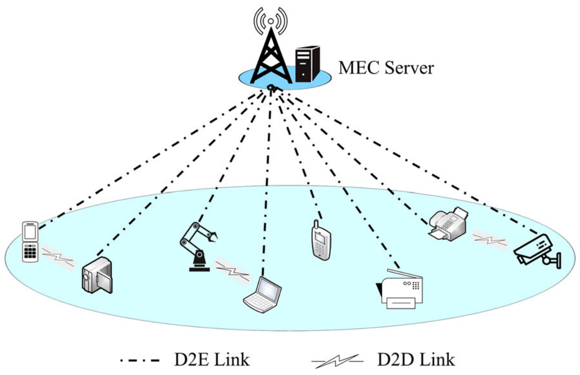

FIGURE 1 Network model with a single base station and multiple devices. D2D, device-to-device; D2E, device-to-edge; MEC, multiaccess edge computing [Color figure can be viewed at wileyonlinelibrary.com]

each device can generate at most one task. Similarly, an SD can only serve at most one TD. Let the triplet $<v_i, c_i, \tau_i>$ represent the task information, where v_i represents the data volume of task i, c_i represents the number of CPU cycles required to calculate a unit of data (bits), and τ_i is the task's delay constraint. Since the amount of data in the calculation result is much smaller than the amount required for calculation, the transmission delay and energy consumption of the computing result are ignored.[38]

The main notations and abbreviations are summarized in Tables 1 and 2, respectively.

3.2 | Communication model

When a task is offloaded to the edge computing server or an SD, the computing data must be transmitted to the task execution node. The TDs communicate with the base station through the cellular links. The data rate r_m^{d2e} is

$$r_m^{\text{d2e}} = B \cdot log_2\left(1 + \frac{p_m^{\text{T}} \cdot d_{m,e}^{-\beta} \left|h_m^{\text{d2e}}\right|^2}{N_0}\right), \qquad \forall m \in \mathcal{M}, \tag{1}$$

TABLE 1 Notions and descriptions

Notation	Description
$\mathcal{N}$	Set of devices
$\mathcal{M}$	Set of task devices
$\mathcal{K}$	Set of service devices
$\mathcal{T}$	Set of time slots, indexed by t
v_i	Input data size of task i
c_i	CPU cycles per bit required of task i
τ_i	Maximum tolerable delay of task i
B	Bandwidth of system
N_o	Background interference power
p_m^{T}	Transmission power of device m
$d_{m,e}$	Distance between device m and eNB
$d_{m,k}$	Distance between device m and device k
$h_{m,e}^{\text{d2e}}$	Channel power gain between device m and eNB
$h_{m,k}^{\text{d2d}}$	Channel power gain between device m and device k
β	Path loss factor
f_{mec}^i	CPU resources assigned to task i by MEC server
f_k^i	CPU resources assigned to task i by device k

Abbreviations: CPU, Central Processing Unit; eNB, enhanced base station; MEC, multiaccess edge computing.

TABLE 2 List of abbreviations

Term	Definition
MEC	Multiaccess edge computing
D2D	Device-to-device
D2E	Device-to-edge
TD	Task device
SD	Service device
RNN	Recurrent neural network
DRL	Deep reinforcement learning
LSTM	Long short-term memory
DDQN	Double deep *Q* network

where B and N_0 represent the bandwidth and noise power, respectively, p_m^{T} is the transmission power of task device m. Further, h_m^{d2e} denotes the channel power gain of the cellular link between device m and the MEC server, $d_{m,e}$ is the distance from device m to the MEC server, and β is the path loss factor.

When the task is offloaded to the SD, TD m and SD k transmit the data through D2D communication link. The data rate $r_{m,k}^{\mathrm{d2d}}$ is

$$r_{m,k}^{\mathrm{d2d}} = B \cdot log_2\left(1 + \frac{p_m^{\mathrm{T}} \cdot d_{m,k}^{-\beta}\left|h_{m,k}^{\mathrm{d2d}}\right|^2}{N_0}\right), \quad \forall m \in \mathcal{M}, \quad \forall k \in \mathcal{K}, \tag{2}$$

where $h_{m,k}^{\mathrm{d2d}}$ is the channel power gain of the D2D link between device m and device k, and $d_{m,k}$ stands for the distance from device m to device k.

3.3 | Computing model

For each task i, it can be executed locally by TD m, or offloaded to either the MEC server or an SD. When a task is executed locally, the task completion latency equals the task computational delay determined by the number of CPU cycles required to execute the task and the locally available computing resources. The local computational delay D_i^{loc} of task i is defined as

$$D_i^{\mathrm{loc}} = \frac{v_i \cdot c_i}{f_m^i}, \quad \forall m \in \mathcal{M}, \tag{3}$$

where v_i represents the data volume of the task i, c_i represents the number of CPU cycles required to calculate a unit data, and f_m^i represents the computing resources allocated by device m to task i.

Deploying a MEC server with rich computing and storage resources in eNB can enable a quick response to the user devices' computing requests, decrease the task completion latency, and reduce backbone network traffic. When TDs generate computing-intensive and delay-sensitive tasks, it is recommended to offload the tasks to the MEC server for execution. In this case, the task completion latency equals the sum of data transmission delay and task computational delay:

$$D_i^{\mathrm{d2e}} = D_{i,\mathrm{trans}}^{\mathrm{d2e}} + D_{i,\mathrm{comp}}^{\mathrm{d2e}}, \tag{4}$$

$$D_{i,\mathrm{trans}}^{\mathrm{d2e}} = \frac{v_i}{r_m^{\mathrm{d2e}}}, \tag{5}$$

$$D_{i,\mathrm{comp}}^{\mathrm{d2e}} = \frac{v_i \cdot c_i}{f_{\mathrm{mec}}^i}, \qquad \forall m \in \mathcal{M}, \tag{6}$$

where $D_{i,\mathrm{trans}}^{\mathrm{d2e}}$ represents the transmission delay of computing data, $D_{i,\mathrm{comp}}^{\mathrm{d2e}}$ is the computational delay of the task, r_m^{d2e} is the D2D communication data rate, and f_{mec}^i denotes the computing resources allocated by the SD to task i.

Compared with the offloading to the MEC server, offloading task to an SD reduces the transmission delay of computing data, the server's computing load, and the pressure on uplink bandwidth. When a TD's computing resources are insufficient, it is feasible to offload the task to an SD for execution through the D2D communication link. Similar to the D2E offloading model, the task completion latency in this scenario can be divided into data transmission delay and task computational delay:

$$D_i^{\mathrm{d2d}} = D_{i,\mathrm{trans}}^{\mathrm{d2d}} + D_{i,\mathrm{comp}}^{\mathrm{d2d}}, \tag{7}$$

$$D_{i,trans}^{d2d} = \frac{v_i}{r_{m,k}^{d2d}}, \tag{8}$$

$$D_{i,\mathrm{comp}}^{\mathrm{d2d}} = \frac{v_i \cdot c_i}{f_k^i}, \qquad \forall m \in \mathcal{M}, \quad \forall k \in \mathcal{K}, \tag{9}$$

where $D_{i,\mathrm{trans}}^{\mathrm{d2d}}$ denotes the transmission delay of computing data, $D_{i,\mathrm{comp}}^{\mathrm{d2d}}$ is the computational delay of task i, $r_{m,k}^{\mathrm{d2d}}$ stands for the data rate of D2D communication, and f_k^i is the computing resources allocated by SD k to task i.

3.4 | Problem description

When TD m generates task i, the node to execute the task is chosen according to the task characteristics and network resources. Let binary vector $\boldsymbol{a_i} = \{a_{i,0}, ..., a_{i,N}\}$ represent the offloading decision of task i. Here, $a_{i,j} \in \{0, 1\}$ represents whether node j should execute task i. In particular, $a_{i,m} = 1$ indicates that the task i is to be executed locally, and $a_{i,0} = 1$ indicates that task i should be offloaded to the MEC server for execution. The task cannot be split, thus

$$\sum_{j=0}^{N} a_{i,j} = 1. \tag{10}$$

The task completion latency of task i generated by TD m is

$$D_i = a_{i,m} \cdot D_i^{\mathrm{loc}} + \sum_{j \in \mathcal{K}} a_{i,j} \cdot D_i^{\mathrm{d2d}} + a_{i,0} \cdot D_i^{\mathrm{d2e}}. \tag{11}$$

Since the MEC server has abundant computing resources, the task computational latency is short when D2E offloading mode is used. However, the transmission delay of calculation data is high. In contrast, when D2D offloading mode is used, the transmission delay of calculation data is short, but the task computational delay is long. By selecting the best offloading nodes for a group of tasks $\mathcal{X} = \{1, ..., X\}$, the total long-term system delay can be minimized under the bandwidth and computing resource constraints. Therefore, the system's optimization goal is formulated as

$$\begin{aligned} \min_{\boldsymbol{a}} \quad & \sum_{t=1}^{T} \sum_{i=1}^{X} D_i(t), \quad t \in \mathcal{T}, \\ \text{s.t.} \quad & C1 : \sum_{i=1}^{X} a_{i,0} \cdot f_{\mathrm{mec}}^{i} \le F_{\mathrm{mec}}, \\ & C2 : a_{ij} \cdot f_j^i \le F_j, \quad i \in \mathcal{X} \text{ and } j \in \mathcal{N}, \\ & C3 : D_i \le \tau_i, \quad i \in \mathcal{X}, \end{aligned} \tag{12}$$

where constraint C_1 denotes that the sum of the task computing demands offloaded to the MEC server must not exceed the total available computing resources of the edge computing server. Constraint C_2 indicates that the task computing demands offloaded to device j shall not be greater than the device's available computing resources. Finally, constraint C_3 means that the actual task completion time shall not exceed the task delay constraint.

4 | DRL-BASED HYBRID OFFLOADING MODEL

The multiconstraint optimization problem in a time-varying system is usually an NP-hard problem.[39] Therefore, this paper models the delay optimization problem of multitask offloading as a Markov decision process and then solves it using DRL. To eliminate the adverse effect of high-dimensional state space on the system model incurred by multitask global decision-making, this study introduces the long short-term memory (LSTM)[40] model into the DRL model. The model extracts the feature information of task and network state, thus improving the convergence speed and stability.

4.1 | Problem formulation

In this section, the delay optimization problem of multitask offloading is systematically modeled, and the problem's state space, action space, and reward function are defined in detail.

4.1.1 | State space

The agent deployed in eNB selects the best execution nodes for the tasks generated by multiple devices according to the task information and current network resources. In time slot t, state $s_t \in \mathcal{S}$ (where $\mathcal{S}$ is the state space) is defined as

$$s_t = \{\boldsymbol{d}(t), \boldsymbol{u}(t), \boldsymbol{v}(t), \boldsymbol{c}(t), \boldsymbol{\tau}(t)\}. \tag{13}$$

In this equation,

- $\boldsymbol{d}(t) = \{\boldsymbol{d}_1(t), ..., \boldsymbol{d}_N(t)\}$ represents the distance from device i to the server and other devices in time slot t. Specifically, $d_{i,0}$ represents the distance between device i and the edge computing server.
- $\boldsymbol{u}(t) = \{u_0(t), ..., u_N(t)\}$ represents the available computing resources in time slot t. $u_0(t)$ represents the server's available computing resources.
- $\boldsymbol{v}(t) = \{v_1(t), ..., v_N(t)\}$ represents the data volume of the task generated by the device in time slot t, in which $v_n(t) = 0$ indicates that device n does not generate any task in time slot.
- $\boldsymbol{c}(t) = \{c_1(t), ..., c_N(t)\}$ represents the calculation density of the task generated by the device in time slot t, that is, the number of CPU cycles required to calculate a unit data.
- $\boldsymbol{\tau}(t) = \{\tau_1(t), ..., \tau_N(t)\}$ represents the delay constraint of the task generated by the device in the time slot t.

4.1.2 | Action space

The agent selects offloading nodes for a group of tasks according to state s_t in time slot t. Action $a_t \in \mathcal{A}$ (where $\mathcal{A}$ is the action space) is defined as

$$a_t = \{\boldsymbol{a}_1(t), ..., \boldsymbol{a}_N(t)\}, \tag{14}$$

where $\boldsymbol{a}_i(t) = [a_{i,0}(t), ..., a_{i,N}(t)]$ is a binary vector representing the offloading position of the task generated by device i. Note that $\sum_{j=0}^{N} a_{i,j}(t) = 1$, meaning that the tasks cannot be split, and all tasks are offloaded to the selected nodes for execution.

4.1.3 | Reward function

This paper's optimization objective is to minimize the system's long-term computational latency. In other words, the system maximizes the long-term reward $\mathcal{R}$ in the reinforcement learning model. The long-term reward is expressed as

$$\mathcal{R} = \sum_{t=1}^{T} r_t, \quad t \in \mathcal{T}. \tag{15}$$

In time slot t, the agent gets an immediate reward r_t after performing an action $a_t \in \mathcal{A}$, and the reward is used to evaluate the offloading decision. The reward function is defined as

$$r_t = \begin{cases} \sum_{i \in \mathcal{X}} D_i^{\text{local}}(t) - D_i(t) & \text{s.t. } C_1, C_2, \text{ and } C_3, \\ -\eta & \text{otherwise,} \end{cases} \tag{16}$$

where $D_i^{\text{local}}(t)$ represents the completion latency of executing the task generated by device i locally, and $D_i(t)$ represents the actual task completion latency after offloading. When the offloading decision complies with the resource and delay constraints, the reward value is the difference between the task's local completion latency and the actual completion latency after offloading (i.e., the benefit of offloading decision). If the offloading decision does not comply with the constraints, the agent receives a penalty value $-\eta$, indicating that the action is unacceptable. Here, η is the design parameter, and $\eta > 0$.

4.2 | DRL model based on RNN

This study introduces RNN into DRL and combines the sequence feature perception ability of the LSTM network with the decision-making ability of the dueling Double Deep *Q* Network (DDQN)[41–43] to build a multitask hybrid offloading model (shown in Figure 2). The core idea of the proposed algorithm is to extract the feature information of task and network state through a bidirectional LSTM and utilize the dueling DDQN for making an offloading decision based on state features. The network model interacts with the environment in real-time, and the generated states, actions, and rewards are stored in the experience pool. The data in the experience pool is sampled randomly, and the sample data is used to construct the objective function and loss function. Then, the network model parameters are trained repeatedly using the functions until the best offloading strategy is obtained. According to the task information and network state, the agent would make the best global offloading decision for multiple tasks in the current time slot, reducing the total long-term system delay.

First, a bidirectional LSTM network is used to extract features of task and network state s_t received by MEC server, which can accelerate the model's convergence. The output o_t of the LSTM network is

$$o_t = \sigma\left(\boldsymbol{W}_o^x \cdot \boldsymbol{x}_t + \boldsymbol{W}_o^h \cdot \boldsymbol{h}_{t-1} + \boldsymbol{b}_o\right), \tag{17}$$

where $\boldsymbol{b}_o$, $\boldsymbol{W}_o^x$, and $\boldsymbol{W}_o^h$ are the weight matrices between offset, input vector, and forgetting gate, respectively. Further, $\boldsymbol{x}_t$ is the input vector, and $\boldsymbol{h}_{t-1}$ is the output of the hidden layer.

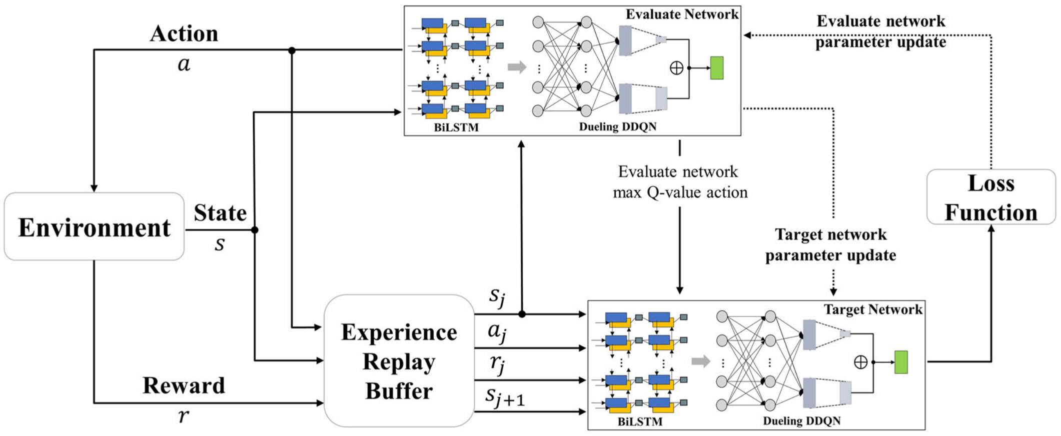

FIGURE 2 Deep reinforcement learning model based on RNN. BiLSTM, bidirectional long short-term memory; DDQN, Double Deep Q Network; RNN, recurrent neural network [Color figure can be viewed at wileyonlinelibrary.com]

Once the bidirectional LSTM extracts feature information o_t, it is processed by the fully connected deep neural network, split, and transferred to two branches. The upper branch represents the value function (denoted $V(s;\theta,\omega)$) about the state, and the lower branch represents the advantage function ($A(s,a;\theta,\varphi)$) about the action. Here, θ is a parameter about the relation between the bidirectional LSTM and the two fully connected layers, ω is the state value network parameter of the upper branch, and φ is the lower branch's action advantage network parameter. By linearly combining the state value function and the action advantage function, the Q estimation value is obtained:

$$Q(s,a;\theta,\omega,\varphi) = V(s;\theta,\omega) + \left(A(s,a;\theta,\varphi) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta,\varphi)\right). \tag{18}$$

The action corresponding to the maximum Q estimation is the task offloading decision. Once the offloading decision is decided, the reward value is calculated by using (16).

Unlike DQN, dueling DDQN no longer directly outputs the maximum action value from the target Q network. Instead, it uses the action corresponding to the estimation network's maximum Q value to calculate target value y'_j in the network

$$y'_j = r_j + \gamma Q'\left(s_{j+1}, \underset{a}{argmax}\ Q(s_{j+1},a;\theta_j,\omega_j,\varphi_j);\theta'_j,\omega'_j,\varphi'_j\right), \tag{19}$$

where γ represents the reward discount factor. The Q estimation network is trained to minimize the loss function. The loss function is defined as

$$Loss(\theta,\omega,\varphi) = \frac{1}{b}\sum_{j=1}^{b}\left(y'_j - Q(s,a;\theta_j,\omega_j,\varphi_j)\right)^2, \tag{20}$$

where b is the number of randomly selected samples.

The obtained state-action and reward information is stored in the experience pool. When the number of samples exceeds the experience pool's capacity, the new experience data overwrites the old one. Once the target number of training samples is reached, the algorithm randomly takes samples to calculate the target value y'_j and loss value in each iteration round. The network parameters are updated via the backpropagation algorithm. The pseudocode of the algorithm is shown in Algorithm 1.

Algorithm 1 Multitask hybrid computing offloading algorithm based on deep reinforcement learning (DRL-MHO)

```
Input: Global state $s_t$ of the system in time slot $t$.
Output: Global offloading decision $a_t$ in time slot $t$.
1: System initialization: initialize the experience pool, network model parameters, and system environment parameters.
2: for each episode: do
3:   Obtain the environment's initial system global state ($s_1$), and extract the state's feature information ($o_1$) using LSTM.
```

```
4:  for each step: do
5:    Select an action ($a_t$) randomly with probability $\epsilon$. Otherwise, select action $a_t = \underset{a}{argmax}\, Q(o_t, a; \theta, \omega, \varphi)$
      with the greatest potential return.
6:    Execute action $a_t$ and calculate the decision reward value ($r_t$) using (16).
7:    Once the system moves to the next state $s_{t+1}$, extract the next state's feature information (i.e., $o_{t+1}$).
8:    Store ($s_t$, $a_t$, $r_t$, $o_{t+1}$) in the experience pool.
9:    Randomly select the experience data set ($s_j$, $a_j$, $r_j$, $s_{j+1}$) from the experience pool batch and calculate the
      target $Q$ value $y_j'$ using (19).
10:   Construct a mean-square error function using (20) and use the error value backpropagation method to
      update the estimation network $Q(\theta, \omega, \varphi)$.
11:   Update the target network parameters at a fixed interval $Q'(\theta', \omega', \varphi') = Q(\theta, \omega, \varphi)$.
12:  end for
13: end for
```

5 | SIMULATION RESULTS AND ANALYSIS

5.1 | Experimental setup

This paper discusses the quasistatic scenario involving a single base station and multiple devices. In each time slot, the device positions remain unchanged, and the channel remains stable. The edge computing server was deployed in the eNB with a coverage radius of 200 m. D2D links share the same band with the cellular links, that is, inband D2D communications.[44] In each time slot, 1–5 devices randomly generated tasks, and the amount of task data and the devices' available computing resources followed the uniform distribution. The proposed model is trained and evaluated by Python 3.7 with Pytorch 1.5.1 on a GPU-based server. The server had one Nvidia GPU of type GTX 1080Ti, and the CPU was Intel Xeno E5-2670 with 64 GB RAM. The FLOPs is 12.8 M, and the time-consumption in the inference phase is 1.93 ms. Adam optimizer is used for training. In the training process of the DRL model, the experience pool capacity was set to 1000, the batch processing quantity b equaled 32, and the reward discount factor γ was 0.9.

This paper adopts the exploration–utilization method to randomly select actions with a probability of ϵ and uses the existing experience to select the action with the highest value with a probability of $1 - \epsilon$. ϵ is defined as

$$\epsilon = \epsilon - ((\epsilon - 0.00001)/10^6), \tag{21}$$

where ϵ is initialized to 0.9 and decreases with the number of iterations. The experimental parameters are shown in Table 3.

In this study, the task processing delay and the device energy consumption of the proposed DRL-MHO algorithm are analyzed and compared with four other task offloading algorithms. In the device energy consumption calculation, only the computational energy consumption required for local task execution and the data transmission energy consumption required for task offloading are considered.[45] The computational energy consumption of the device required for local task execution can be expressed as[46]

$$E_m^{\mathrm{loc}} = D_m^{\mathrm{loc}} \cdot p_m^{\mathrm{C}}. \tag{22}$$

TABLE 3 Experimental parameters

Parameter	Value
Coverage radius of base station	200 m
The server's computing resources, F_{MEC}	6 GHz
The device's computing resources, F_m	[1, 1.5] GHz
The device's maximum transmission power, p_m^{T}	250 mV
Floating-point calculation power of the device, p_m^{C}	500 mV
Maximum D2D communication distance, ξ	50 m
Bandwidth, B	2 MHz
Noise, N_0	−174 dBm
Task data volume, v_m	[200, 800] Kbit
Task calculation density, c_m	1000 cycles/bit
Task delay constraint, τ_m	600 ms

Abbreviations: D2D, device-to-device; MEC, multiaccess edge computing.

The data transmission energy consumption can be expressed as

$$E_m^{\text{tran}} = \frac{v_m}{r_m} \cdot p_m^{\text{T}}, \tag{23}$$

where D_m^{loc} represents the computational delay of local task execution, p_m^{C} denotes the power consumption of the user device per unit time, v_m is the data volume of the task generated by the user device, r_m is the data transmission rate of the user device, and p_m^{T} denotes the data transmission power of the user device

The other algorithms considered in this study are:

(1) *Local computing* (*LC*): All tasks generated by the devices are processed locally.
(2) *Offloading to a device* (*D2D*): The tasks generated by the devices can only be processed locally or offloaded to the adjacent devices for processing.
(3) *Offloading to the edge server* (*D2E*): All tasks generated by the devices are offloaded to the edge computing server for execution.
(4) *DQN-based hybrid offloading algorithm* (*VECN*)[47]: Hybrid offloading is performed jointly by the local fixed edge server and vehicular mobile servers. During the offloading process, the algorithm only optimizes the completion latency of individual tasks without considering the impact of user decisions on the overall system performance.

5.2 | Experimental result

The performance of the proposed DRL-MHO model is first compared with the performance of two commonly used DRL models, namely, DDQN and dueling DDQN. DDQN solves the overestimation problem of DQN by separating selection from evaluation. Since dueling DDQN divides the *Q* value into state value and action advantage, its estimated value function is more accurate, potentially leading to better strategy obtained through learning. As a synthesis of RNN and DRL model, DRL-MHO uses bidirectional LSTM to extract the state's feature information and takes the feature information as the input of dueling DDQN, improving the convergence speed and model stability (see Figure 3A). In addition, DRL-MHO can minimize the system's long-term computational delay, as shown in Figure 3B.

Replacing the deep neural network with an RNN necessarily increases the overall model parameters and complexity, thus prolonging the iteration time. Nevertheless, it enables achieving a significant improvement in the overall convergence speed and model stability at the expense of a small amount of memory. Indeed, as shown in Table 4, DRL-MHO has slightly more parameters and a longer single iteration time than DDQN and dueling DDQN, but its convergence speed is much faster than DDQN and dueling DDQN.

5.2.1 | System delay

This study proposes a global offloading decision mechanism based on DRL for the multitask hybrid offloading scenario with the optimization goal of reducing the long-term total system delay. It can be seen from Figure 4A that two device-edge hybrid offloading methods are superior to the single-node offloading methods in terms of total system delay, while the DRL-MHO's performance is significantly better than that of the other four methods. Compared with the other four methods, the total system delay can be reduced by approximately 40.8%, 31.1%, 24.2%, and 15.3%, respectively. Since the number of devices generating tasks in each time slot, as well as the amount of task data, are stochastic, the average computational delay of each time slot fluctuates. In most time slots, the average computational delay of the DRL-MHO algorithm is the lowest (Figure 4B). Nevertheless, in several instances, vehicle edge computing and network (VECN) and D2E achieve the lowest average delay. This result stems from this study's focus on specific multitask global offloading scenarios. Namely, DRL-MHO's optimization goal is to reduce the system's long-term total computational delay rather than the computational

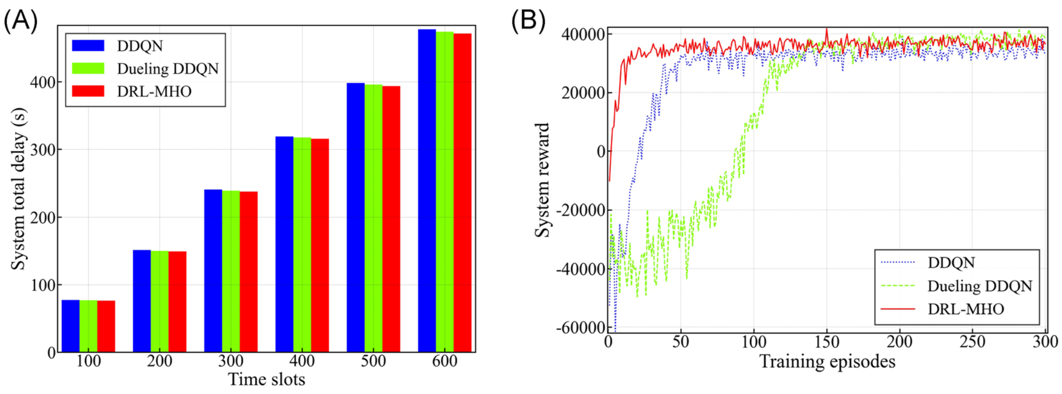

FIGURE 3 Performance comparison with deep reinforcement learning models. (A) Comparison of model convergence speed and (B) comparison of the total system delay. DDQN, Double Deep Q Network; DRL, deep reinforcement learning; MHO, multitask hybrid computing offloading [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 4 Performance comparison with deep reinforcement learning models

Performance parameter	DDQN	Dueling DDQN	DRL-MHO
Number of network layers	6	6	6
Model parameter quantity	12,872,572	12,873,597	14,337,661
Iteration time (s)	18.71	19.28	21.29
Number of iterations	60	130	10
Decision performance	Poor	Suboptimal	Optimal

Abbreviations: DDQN, Double Deep Q Network; DRL, deep reinforcement learning; MHO, multitask hybrid computing offloading.

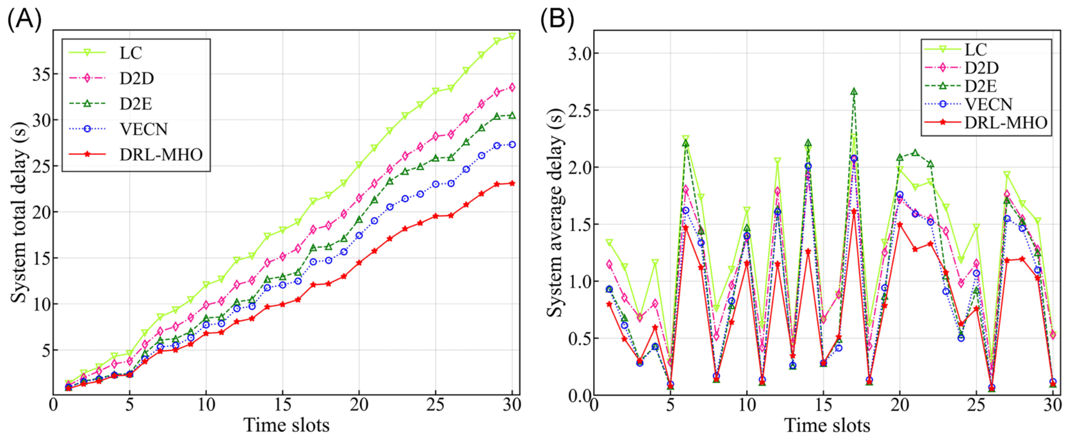

FIGURE 4 Comparison of total and average delays of the system. (A) Total system computational delay and (B) average system computational delay. D2D, device-to-device; D2E, device-to-edge; DRL, deep reinforcement learning; LC, local computing; MHO, multitask hybrid computing offloading; VECN, vehicle edge computing and network [Color figure can be viewed at wileyonlinelibrary.com]

delay of individual tasks. Thus, when the number of generated tasks is relatively small in specific time slots, the average computational delays of VECN and D2E are lower than that of DRL-MHO.

Figure 5 shows the impact of the number of tasks on the system delay. When the number of tasks is small, the D2E's delay is the lowest because the MEC server has sufficient computing resources and uplink bandwidth under such conditions. However, as the number of tasks increases, less and less computing and bandwidth resources are available for each task, rapidly increasing the D2E's delay. In contrast, DRL-MHO adopts the global hybrid offloading approach to deal with multiple tasks and considers the impact of the offloading decisions of individual tasks on system resource allocation. Its optimization goal is to reduce the total system delay rather than the computational delay of individual tasks. Therefore, with the increase in the number of tasks, the advantages of DRL-MHO become evident.

Figure 6 shows the impact of the amount of task data on the total computational delay of the system. The system delay of every offloading method increases with the amount of task data. When the system computing and bandwidth resources are limited, the larger the task

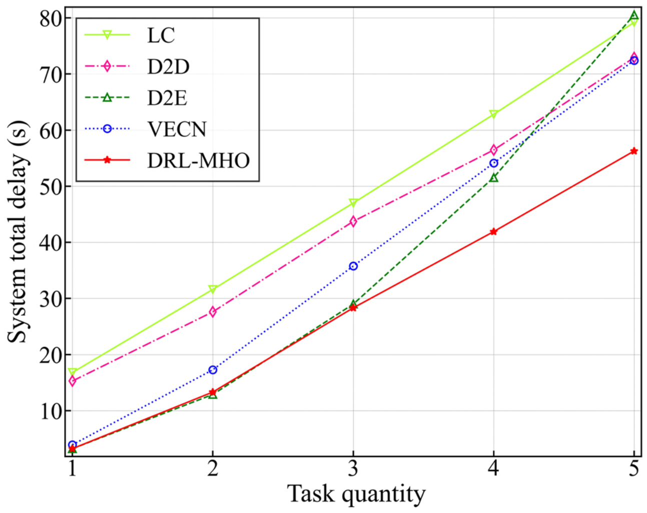

FIGURE 5 Impact of task quantity on system delay. D2D, device-to-device; D2E, device-to-edge; DRL, deep reinforcement learning; LC, local computing; MHO, multitask hybrid computing offloading; VECN, vehicle edge computing and network [Color figure can be viewed at wileyonlinelibrary.com]

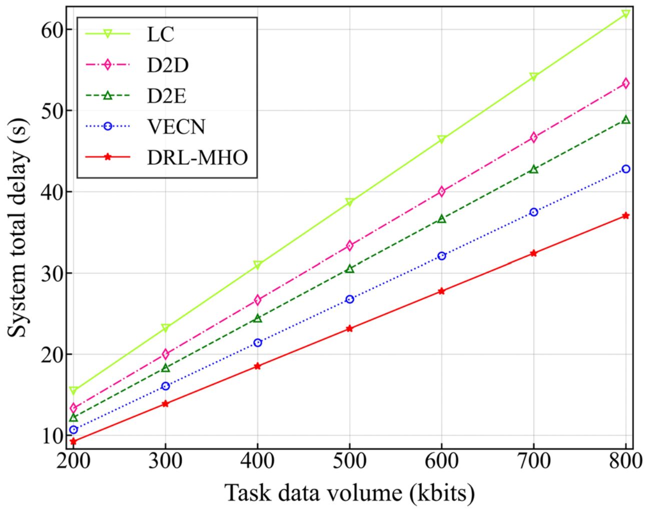

FIGURE 6 Impact of task data volume on system delay. D2D, device-to-device; D2E, device-to-edge; DRL, deep reinforcement learning; LC, local computing; MHO, multitask hybrid computing offloading; VECN, vehicle edge computing and network [Color figure can be viewed at wileyonlinelibrary.com]

data, the longer the computational delay and transmission delay. The performance advantage of DRL-MHO increases with the amount of task data, thus further verifying the DRL-MHO's strong generalization ability when faced with different tasks.

Figure 7 shows the impact of the edge server's available computing resources on the task processing delay of the system. With the increase in the server's computing resources, the total delays of D2E and two hybrid offloading mechanisms decrease. When the server computing resources are in short supply, the total system delay of D2E is much higher than those of other

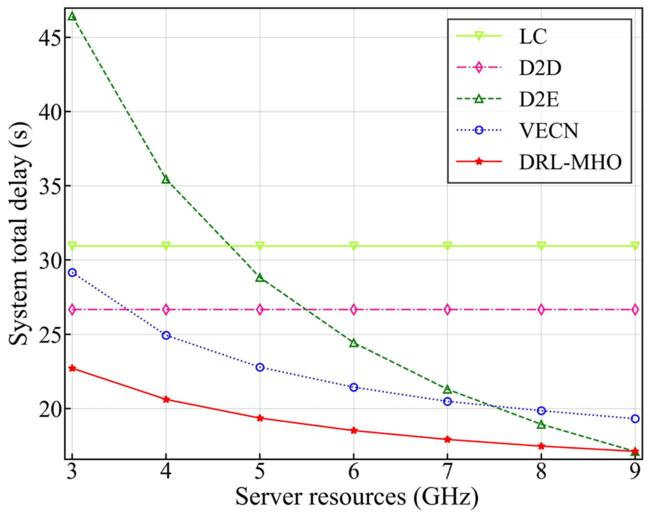

FIGURE 7 Impact of server resources on system delay. D2D, device-to-device; D2E, device-to-edge; DRL, deep reinforcement learning; LC, local computing; MHO, multitask hybrid computing offloading; VECN, vehicle edge computing and network [Color figure can be viewed at wileyonlinelibrary.com]

mechanisms. When the server is rich in resources, it can meet the computing needs of multiple tasks. At this time, offloading all tasks to the server for processing can minimize the system delay. The multitask hybrid offloading algorithm proposed in this paper selects the appropriate offloading nodes according to the network resources, achieving the lowest total system delay regardless of whether the server computing resources are abundant or in short supply.

5.2.2 | Energy consumption

Since mobile devices are sensitive to energy consumption, this paper considers only the energy consumption of the devices in the system. The system energy consumption is divided into two parts: computational energy consumption and data transmission energy consumption. Most of the system energy consumption is the computational energy consumption, which increases with the increase of computational delay. As seen in Figure 8A, the total energy consumption of D2E is the smallest. This is because in the scenario of devices offloading the tasks to the edge computing server for execution, the system energy consumption only consists of data transmission energy consumption. Compared with the other four offloading mechanisms, DRL-MHO has the lowest total system delay and the second-lowest device energy consumption (only higher than D2E), demonstrating that it can effectively improve the user experience and system performance. Figure 8B shows the average energy consumption per time slot. In time slots with large numbers of tasks, the energy consumption of DRL-MHO is only higher than that of D2E.

Figures 9 and 10 show the effects of the number of tasks and the amount of task data on the system energy consumption. Similar to the delay, the advantages of DRL-MHO in reducing energy consumption become more pronounced with the increase in the number of tasks and the amount of task data.

Designed to offload multiple tasks simultaneously, DRL-MHO optimizes system performance from a global perspective and takes into account the impact of individual devices' task

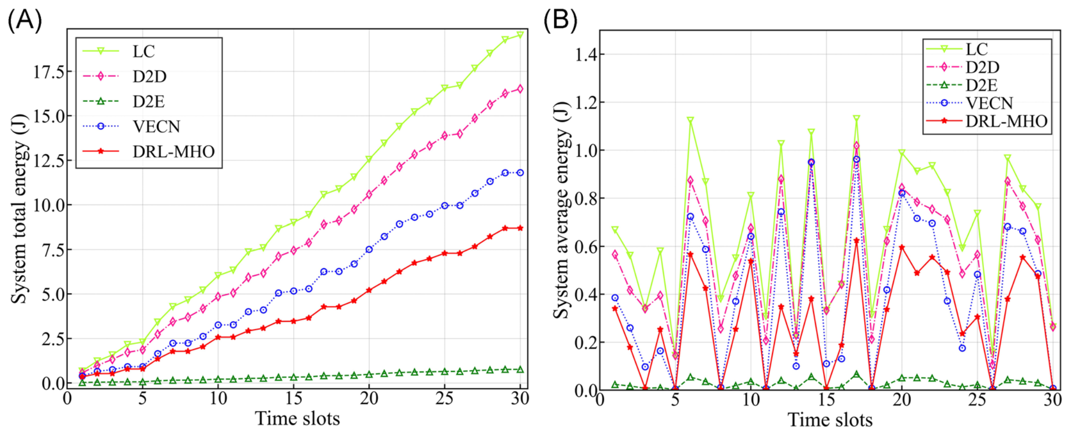

FIGURE 8 Total energy consumption and average energy consumption of the system. (A) Total energy consumption of the system and (b) average energy consumption of the system. D2D, device-to-device; D2E, device-to-edge; DRL, deep reinforcement learning; LC, local computing; MHO, multitask hybrid computing offloading; VECN, vehicle edge computing and network [Color figure can be viewed at wileyonlinelibrary.com]

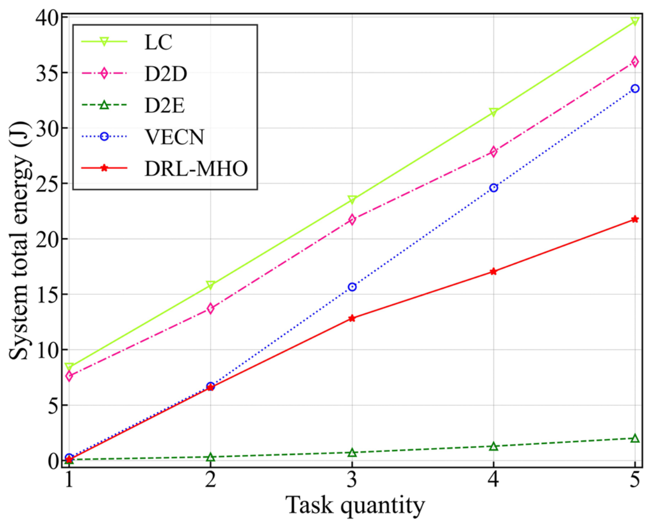

FIGURE 9 Impact of task quantity on energy consumption. D2D, device-to-device; D2E, device-to-edge; DRL, deep reinforcement learning; LC, local computing; MHO, multitask hybrid computing offloading; VECN, vehicle edge computing and network [Color figure can be viewed at wileyonlinelibrary.com]

offloading on system resource allocation. DRL-MHO is superior to the single-node offloading mechanisms in terms of delay and energy consumption. VECN, another hybrid offloading mechanism based on DRL, offloads multiple tasks one by one. It only optimizes the delay of the current task, ignoring the impact of the offloading decision made for the current task on other tasks. Therefore, the performance of VECN is lower than that of DRL-MHO. Especially in the multiaccess and multitask scenario, the long-term system delay of DRL-MHO is 20% lower than that of VECN.

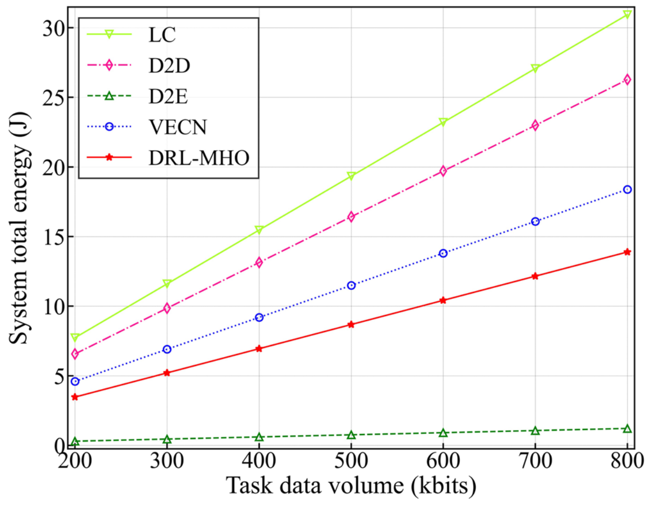

FIGURE 10 Impact of task data volume on energy consumption. D2D, device-to-device; D2E, device-to-edge; DRL, deep reinforcement learning; LC, local computing; MHO, multitask hybrid computing offloading; VECN, vehicle edge computing and network [Color figure can be viewed at wileyonlinelibrary.com]

6 | CONCLUSION

This paper proposes DRL-MHO, an MHO model based on DRL for the multiaccess and multitask edge computing scenario. The proposed algorithm makes global offloading decisions for multiple tasks. It offloads the tasks generated by a group of devices to the edge computing server and adjacent devices, thus reducing the long-term overall system delay. By introducing the bidirectional LSTM model into DRL, the proposed model extracts task and network states' feature information to improve the convergence speed and model stability. In the simulation experiment, the DRL-MHO model's performance was compared with that of algorithms of different types, including local execution, binary offloading, and device-edge hybrid offloading algorithms. The experimental results demonstrate that the proposed algorithm effectively reduces the long-term task completion latency of the system and device energy consumption, thus improving the user experience.

The proposed DRL-MHO algorithm models the task offloading as a single-objective optimization problem, focuses on minimizing completion latency of computing task in edge computing scenario. It cannot meet differentiated requirements of users in a heterogeneous network. The future research will target the cloud-edge-device hybrid offloading scenario, focusing on computing offloading for tasks with differentiated quality of service.

ACKNOWLEDGMENTS

This study was supported by the National Key Research and Development Program of China (2019YFB1804403 and 2018YFB1802200), the National Natural Science Foundation of China (61972104, 61902080, 62002072, and 61702120), the Special Projects in Key Areas of Guangdong Province (2019B010118001), the Science and Technology Project in Guangzhou (201803010081), Foshan Science and Technology Innovation Project, China (2018IT100283), Guangzhou Key Laboratory (202102100006), Science and Technology Program of Guangzhou, China (202002020035), and the Research Projects in Guangdong Province (2021ZDJS026).

ORCID

Yan Liu https://orcid.org/0000-0002-8257-2701

REFERENCES

1. Kaur K, Garg S, Aujla GS, Kumar N, Rodrigues JJPC, Guizani M. Edge computing in the industrial internet of things environment: software-defined-networks-based edge-cloud interplay. *IEEE Commun Mag*. 2018; 56(2):44-51. doi:10.1109/MCOM.2018.1700622
2. Kim J, Yun U, Kim H, et al. Average utility driven data analytics on damped windows for intelligent systems with data streams. *Int J Intell Syst*. 2021;36(10):5741-5769. doi:10.1002/int.22528
3. Mach P, Becvar Z. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tutorials*. 2017;19(3):1628-1656. doi:10.1109/COMST.2017.2682318
4. Khan AuR, Othman M, Madani SA, Khan SU. A survey of mobile cloud computing application models. *IEEE Commun Surv Tutorials*. 2014;16(1):393-413. doi:10.1109/SURV.2013.062613.00160
5. Wang Y, Chen IR, Wang DC. A survey of mobile cloud computing applications: perspectives and challenges. *Wireless Pers Commun*. 2015;80(4):1607-1623. doi:10.1007/s11277-014-2102-7
6. Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. *Commun ACM*. 2010;53(4):50-58. doi:10.1145/1721654.1721672
7. Wang S, Zhang X, Zhang Y, Wang L, Yang J, Wang W. A survey on mobile edge networks: convergence of computing, caching and communications. *IEEE Access*. 2017;5:6757-6779. doi:10.1109/ACCESS.2017.2685434
8. Xie R, Lian X, Jia Q. Survey on computation offloading in mobile edge computing. *J Commun*. 2018;39(11): 138-155. doi:10.11959/j.issn.1000-436x.2018215
9. Sabella D, Vaillant A, Kuure P, Rauschenbach U, Giust F. Mobile-edge computing architecture: the role of MEC in the internet of things. *IEEE Consum Electron Mag*. 2016;5(4):84-91. doi:10.1109/MCE.2016.2590118
10. Chen H, Yu J, Zhou H, Zhou T, Liu F, Cai Z. SmartStore: a blockchain and clustering based intelligent edge storage system with fairness and resilience. *Int J Intell Syst*. 2021;36(9):5184-5209. doi:10.1002/int.22509
11. Zhang K, Gui X, Ren D. Survey on computation offloading and content caching in mobile edge networks. *J Software*. 2019;30(8):2491-2516. doi:10.13328/j.cnki.jos.005861
12. Zhao X, Peng J, You W. A privacy-aware computation offloading method based on Lyapunov optimization. *J Electron Inf Technol*. 2020;42(3):704-711. doi:10.11999/JEIT190170
13. Abdel-Basset M, Mohamed R, Chakrabortty RK, Ryan MJ. IEGA: an improved elitism-based genetic algorithm for task scheduling problem in fog computing. *Int J Intell Syst*. 2021;36(9):4592-4631. doi:10.1002/int.22470
14. Lu H, He X, Du M, Ruan X, Sun Y, Wang K. Edge QoE: computation offloading with deep reinforcement learning for internet of things. *IEEE Internet Things J*. 2020;7(10):9255-9265. doi:10.1109/JIOT.2020.2981557
15. Li H, Xu H, Zhou C, Lü X, Han Z. Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment. *IEEE Trans Veh Technol*. 2020;69(9):10214-10226. doi:10.1109/TVT.2020.3003898
16. Ning Z, Wang X, Rodrigues JJPC, Xia F. Joint computation offloading, power allocation, and channel assignment for 5G-enabled traffic management systems. *IEEE Trans Ind Inf*. 2019;15(5):3058-3067. doi:10.1109/TII.2019.2892767
17. Chatzopoulos D, Bermejo C, Haq Eu, Li Y, Hui P. D2D task offloading: a dataset-based Q&A. *IEEE Commun Mag*. 2019;57(2):102-107. doi:10.1109/MCOM.2018.1700873
18. Chatzopoulos D, Ahmadi M, Kosta S, Hui P. Have you asked your neighbors? A hidden market approach for device-to-device offloading. In: *Proceedings of the 2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE; 2016:1-9.
19. Fei Z, Wang Y, Sun R, Liu Y. Delay-oriented task scheduling and bandwidth allocation in fog computing networks. In: *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE; 2019:1-6.
20. Huang L, Bi S, Zhang YJA. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Trans Mobile Comput*. 2020;19(11):2581-2593. doi:10.1109/TMC.2019.2928811

21. Lei L, Xu H, Xiong X, Zheng K, Xiang W, Wang X. Multiuser resource control with deep reinforcement learning in IoT edge computing. *IEEE Internet Things J*. 2019;6(6):10119-10133. doi:10.1109/JIOT.2019.2935543
22. Li J, Gao H, Lv T, Lu Y. Deep reinforcement learning based computation offloading and resource allocation for MEC. In: *Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE; 2018:1-6.
23. Zhao H, Zhang T, Chen Y. Task distribution offloading algorithm of vehicle edge network based on DQN. *J Commun*. 2020;41(10):172-178.
24. Guo K, Gao R, Xia W, Quek TQS. Online learning based computation offloading in MEC systems with communication and computation dynamics. *IEEE Trans Commun*. 2021;69(2):1147-1162. doi:10.1109/TCOMM.2020.3038875
25. Liu C, Tang F, Hu Y, Li K, Tang Z, Li K. Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach. *IEEE Trans Parallel Distrib Syst*. 2021;32(7):1603-1614. doi:10.1109/TPDS.2020.3046737
26. Wu Y, Shi B, Qian LP, Hou F, Cai J, Shen XS. Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs. *IEEE Trans Ind Inf*. 2020;16(7):4811-4822. doi:10.1109/TII.2019.2944839
27. Sahni Y, Cao J, Yang L, Ji Y. Multi-hop multi-task partial computation offloading in collaborative edge computing. *IEEE Trans Parallel Distrib Syst*. 2021;32(5):1133-1145. doi:10.1109/TPDS.2020.3042224
28. Chen Z, Zhang L, Pei Y, Jiang C, Yin L. NOMA-based multi-user mobile edge computation offloading via cooperative multi-agent deep reinforcement learning. *IEEE Trans Cogn Commun Networking*. 2021;1:1-4. doi:10.1109/TCCN.2021.3093436
29. Feng J, Zhao L, Du J, Chu X, Yu FR. Computation offloading and resource allocation in D2D-enabled mobile edge computing. In: *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*. IEEE; 2018:1-6.
30. Seng S, Li X, Luo C, Ji H, Zhang H. A D2D-assisted MEC computation offloading in the blockchain-based framework for UDNs. In: *Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE; 2019:1-6.
31. Yu S, Dab B, Movahedi Z, Langar R, Wang L. A socially-aware hybrid computation offloading framework for multi-access edge computing. *IEEE Trans Mobile Comput*. 2020;19(6):1247-1259. doi:10.1109/TMC.2019.2908154
32. Peng J, Qiu H, Cai J, Xu W, Wang J. D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC. *IEEE Trans Wireless Commun*. 2021;20(8):4858-4873. doi:10.1109/TWC.2021.3062616
33. He Y, Ren J, Yu G, Cai Y. D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. *IEEE Trans Wireless Commun*. 2019;18(3):1750-1763. doi:10.1109/TWC.2019.2896999
34. Li G, Chen M, Wei X, Qi T, Zhuang W. Computation offloading with reinforcement learning in D2D-MEC network. In: *Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE; 2020:69-74.
35. Jiang C, Cao T, Guan J. Intelligent task offloading and collaborative computation over D2D communication. *China Commun*. 2021;18(3):251-263. doi:10.23919/JCC.2021.03.020
36. Cheng Y, Liang C, Chen Q, Yu R. Energy-efficient D2D-assisted computation offloading in NOMA-enabled cognitive networks. *IEEE Trans Veh Technol*. 2021;70(12):13441-13446. doi:10.1109/TVT.2021.3093892
37. Chai R, Lin J, Chen M, Chen Q. Task execution cost minimization-based joint computation offloading and resource allocation for Cellular D2D MEC systems. *IEEE Syst J*. 2019;13(4):4110-4121. doi:10.1109/JSYST.2019.2921115
38. Chen X, Jiao L, Li W, Fu X. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans Networking*. 2016;24(5):2795-2808. doi:10.1109/TNET.2015.2487344
39. Zhang H, Yang Y, Huang X, Fang C, Zhang P. Ultra-low latency multi-task offloading in mobile edge computing. *IEEE Access*. 2021;9:32569-32581. doi:10.1109/ACCESS.2021.3061105
40. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9(8):1735-1780. doi:10.1162/neco.1997.9.8.1735

41. Volodymyr Mnih DSAARJV. Human-level control through deep reinforcement learning. *Nature*. 2015; 518(7540):529-540.
42. Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. AAAI Press; 2016:2094-2100.
43. Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: Balcan MF, Weinberger KQ, eds. *Proceedings of the 33rd International Conference on Machine Learning*. Vol 48. PMLR; 2016:1995-2003.
44. Asadi A, Wang Q, Mancuso V. A survey on device-to-device communication in cellular networks. *IEEE Commun Surv Tutorials*. 2014;16(4):1801-1819. doi:10.1109/COMST.2014.2319555
45. Saleem U, Liu Y, Jangsher S, Tao X, Li Y. Latency minimization for D2D-enabled partial computation offloading in mobile edge computing. *IEEE Trans Veh Technol*. 2020;69(4):4472-4486. doi:10.1109/TVT.2020.2978027
46. Tang L, Xiao J, Wei Y. Joint resource allocation algorithms based on mixed cloud/fog computing in vehicular network. *J Electron Inf Technol*. 2020;42(8):1926-1933. doi:10.11999/JEIT190306
47. Liu Y, Yu H, Xie S, Zhang Y. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Trans Veh Technol*. 2019;68(11):11158-11168. doi:10.1109/TVT.2019.2935450

How to cite this article: Cai J, Fu H, Liu Y. Deep reinforcement learning-based multitask hybrid computing offloading for multiaccess edge computing. *Int J Intell Syst*. 2022;1-23. doi:10.1002/int.22841