sensors

MDPI

*Article*

# Deep Reinforcement Learning for Computation Offloading and Resource Allocation in Unmanned-Aerial-Vehicle Assisted Edge Computing

**Shuyang Li, Xiaohui Hu * and Yongwen Du**

School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China; 0619680@stu.lzjtu.edu.cn (S.L.); duyongwen@mail.lzjtu.cn (Y.D.)
* Correspondence: huxiaohui@mail.lzjtu.cn

**Abstract:** Computation offloading technology extends cloud computing to the edge of the access network close to users, bringing many benefits to terminal devices with limited battery and computational resources. Nevertheless, the existing computation offloading approaches are challenging to apply to specific scenarios, such as the dense distribution of end-users and the sparse distribution of network infrastructure. The technological revolution in the unmanned aerial vehicle (UAV) and chip industry has granted UAVs more computing resources and promoted the emergence of UAV-assisted mobile edge computing (MEC) technology, which could be applied to those scenarios. However, in the MEC system with multiple users and multiple servers, making reasonable offloading decisions and allocating system resources is still a severe challenge. This paper studies the offloading decision and resource allocation problem in the UAV-assisted MEC environment with multiple users and servers. To ensure the quality of service for end-users, we set the weighted total cost of delay, energy consumption, and the size of discarded tasks as our optimization objective. We further formulate the joint optimization problem as a Markov decision process and apply the soft actor–critic (SAC) deep reinforcement learning algorithm to optimize the offloading policy. Numerical simulation results show that the offloading policy optimized by our proposed SAC-based dynamic computing offloading (SACDCO) algorithm effectively reduces the delay, energy consumption, and size of discarded tasks for the UAV-assisted MEC system. Compared with the fixed local-UAV scheme in the specific simulation setting, our proposed approach reduces system delay and energy consumption by approximately 50% and 200%, respectively.

**Keywords:** unmanned aerial vehicle; edge computing; computation offloading; resource allocation; soft actor–critic



**Citation:** Li, S.; Hu, X.; Du, Y. Deep Reinforcement Learning for Computation Offloading and Resource Allocation in Unmanned-Aerial-Vehicle Assisted Edge Computing. *Sensors* **2021**, *21*, 6499. https://doi.org/10.3390/s21196499

Academic Editor: Andrey V. Savkin

Received: 27 August 2021
Accepted: 25 September 2021
Published: 29 September 2021



## 1. Introduction

In the past decade, the exponential growth and diversity of Internet of Things (IoT) devices have changed the way we live [1]. Advances in 5G technology and the IoT have made many emerging applications possible, such as autonomous driving, smart cities, virtual reality (VR)/augmented reality (AR), real-time video analysis, and cloud gaming. Nevertheless, the traditional cloud-based computing paradigm is not suitable for those IoT terminal devices with limited computing and battery resources. The emergence of mobile edge computing (MEC) technology is expected to improve this situation. The MEC server deployed at the edge of the access network can provide terminal devices with computing and communication resources, bringing many benefits, such as reducing computing workload, delay, network congestion, and energy consumption. Traditional MEC servers are usually deployed on cellular base stations (BS) or Wi-Fi access points (AP). However, considering the construction cost and the gradual infrastructure update, not all BSs and APs can deploy edge servers. Therefore, mobile platforms, such as vehicles and UAVs are regarded as alternative candidates for MEC servers.

### 1.1. Motivation and Related Work

The existing literature has shown the great potential and capability of edge computing, which usually takes delay, energy consumption, system costs, etc., as the optimization objective [2–10]. However, all those studies assume that the stable wired or wireless communication link is established with sufficient bandwidth between the end-user and the distributed edge resources deployed fixedly. The existing computation offloading approaches are challenging to apply to specific scenarios, such as the dense distribution of end-users and the sparse distribution of network infrastructure.

With the advantages of flexible deployment, high mobility, strong line-of-sight (LOS) connection, and hover capability, UAVs are expected to play a critical role in wireless networks. Specifically, since UAVs can be deployed freely and flexibly in three-dimensional space, direct LOS communication with any terminal device can be easily established. This advantage allows UAVs to be used as relay nodes in wireless networks to improve communication between end-users. In addition, with the evolution of the UAV and chip industry, more computing and storage resources could be placed on UAVs, making it possible for UAVs to provide value-added computing services. UAVs not only play an essential role in the military domains but are also widely used in the civilian domains [11–14]. Due to its compelling features, UAVs as a kind of auxiliary computing and communication entity, which has been considered in the MEC technology. UAV-assisted edge computing technology could effectively provide computing and communication support for end-users in the special scenarios mentioned above. In UAV-assisted edge computing systems, the UAV can adjust its position in time according to the dynamic communication environment, ensuring that a reliable communication link can be established between the end-user and the MEC server. So far, tons of studies have shown the feasibility of UAV-assisted edge computing offloading technology.

However, in most previous studies [15,16], the computing capacity provided by UAVs was ignored in UAV-assisted wireless networks, and the communication capacity was mainly considered. Recent studies have begun to consider the computing resources of UAVs. Ref. [17–20] have only considered the communication and computing interaction between the two types of entities, which is the end-user offload part of the computing tasks to the UAV through wireless communication. Compared with UAVs, MEC servers have more computing and storage resources and are not limited by battery capacity. However, existing studies have rarely considered letting UAVs and MEC servers cooperate in providing end-user services. As far as we know, UAV-assisted MEC systems involving end-users, UAVs, and MEC servers have rarely been studied.

Reinforcement learning is a control-theoretic trial-and-error learning method [21]. The agent interacts with the environment and makes decisions through feedback from the environment. Existing studies have proved that reinforcement learning can handle offloading decision-making in the MEC environment well. Huang et al. [22] have proposed a DQN-based approach for computation offloading and resource allocation, which minimize the overall offloading cost of the MEC environment. Yang et al. [23] have proposed a DQN-based optimization approach for task scheduling in the UAV-assisted MEC environment, which could improve the efficiency of the task execution in each UAV. As far as we know, most DRL-based computation offloading approaches use the DQN algorithm. The selection of DRL algorithms depends highly on the dimension of state space and action space. Compared with the traditional Q-learning algorithm, the DQN algorithm applies the neural network to approximate the Q-value, which could handle high-dimensional state problems well. However, the DQN algorithm cannot handle high-dimensional action problems.

### 1.2. Main Contributions

To fill the previous studies, we envision a UAV-assisted MEC system consisting of multiple edge servers, multiple end-users, and a UAV. The UAV and MEC servers in the MEC system cooperatively work to provide computing services for end-users. With the help of the UAVs, the end-user can offload the tasks to the MEC server outside of its

communication range and execute tasks by the MEC servers. We also consider that the computing task is partially offloaded, which is different from most existing studies' binary offloading model and is closer to the actual situation [20]. We take the minimization of the weighted total cost of delay, energy consumption, and the size of discarded tasks as the optimization objective and further formulate the offloading decision problem as a Markov decision process. To this end, we propose a dynamic computation offloading approach based on the soft actor–critic (SAC) DRL algorithm. The SAC algorithm introduces entropy into the traditional actor–critic algorithm, improves the decision-making performance and obtains the global optimal policy. Numerical simulation results have proved the effectiveness of our proposed SAC-based dynamic computing offloading (SACDCO) algorithm compared with other baseline schemes. The differences between our work and the existing literature are summarized in Table 1.

The remainder of this paper is organized as follows. Section 2 describes the system model and problem formulation. Section 3 introduces the UAV-assisted edge computation offloading approach we proposed. The performance evaluation of our proposed approach is achieved through a series of simulations, and numerical results are given in Section 4. Section 5 summarizes the paper.

**Table 1.** Comparison between our work and the existing literature. (✓) indicates that the topic is covered.

| Reference | Communication-Only | Communication and Computation | EU-UAV | EU-UAV-MEC | Partial Offloading | RL Algorithm | Optimization Objective |
|---|---|---|---|---|---|---|---|
| [15] | ✓ | | | | | | Throughput |
| [16] | ✓ | | | | | | Throughput |
| [17] | | ✓ | ✓ | | ✓ | | Delay |
| [18] | | ✓ | ✓ | | | | Energy consumption |
| [19] | | ✓ | ✓ | | | | Computation rate |
| [20] | | ✓ | ✓ | | | | Energy consumption |
| [22] | | ✓ | ✓ | | | DQN | The cost of energy, computation, and delay |
| [23] | | ✓ | ✓ | | | DQN | Load balance |
| [24] | | ✓ | ✓ | | ✓ | DQN | System utility |
| [25] | | ✓ | ✓ | | ✓ | Actor–Critic | Average response time |
| [26] | | ✓ | | ✓ | | DQN | System reward |
| Our work | | ✓ | | ✓ | ✓ | Soft Actor–Critic | The cost of delay, energy, and discarded tasks |

## 2. System Model and Problem Formulation

### *2.1. System Model*

The UAV-assisted MEC system is composed of multiple edge servers, multiple end-users, and a UAV, which is shown in Figure 1. We consider a set of end-users, and each end-user periodically executes compute-intensive and delay-sensitive tasks during the decision episode. Due to signal congestion and the limited communication distance of the end-user, stable wireless communication cannot be established between the end-user and the MEC server. The UAV is equipped with antennas to communicate with end-users and MEC servers in the coverage area. In UAV-assisted MEC systems, end-users can offload computing tasks to UAVs. Compared with end-users, UAV has stronger computing power, but it still cannot compare with the computing power of MEC servers. Considering the limited battery capacity and computing power of the UAV, if the UAV cannot complete the computing task well, it will further consider offloading the computing task to the MEC server in the distance. The follow-me cloud (FMC) [27] controller is used in our proposed UAV-assisted MEC system, which could obtain the global information of end-users, MEC servers, and the UAV. Therefore, the proposed dynamic computing offloading algorithm is executed on the FMC controller.

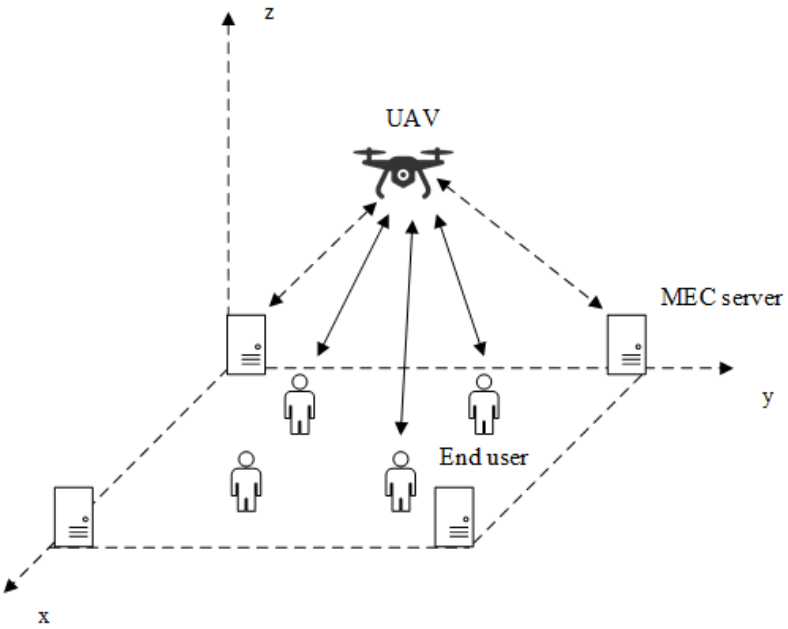


**Figure 1.** The architecture of UAV-assisted mobile edge computing system.

Without loss of generality, a set of end-users is denoted by $\mathcal{N} = \{1, 2, \cdots, n, \cdots, N\}$, a set of MEC servers is denoted by $\mathcal{S} = \{1, 2, \cdots, s, \cdots, S\}$, and the UAV is denoted by $\mathcal{U} = \{u\}$. The entire decision episode is divided into multiple time slots, where $\mathcal{T} = \{1, 2, \cdots, t, \cdots, T\}$ denotes their corresponding set. The UAV stays at a fixed altitude $h_u(t) = H, \forall t \in \mathcal{T}$. We define the three dimensional Cartesian coordinates of the UAV as $L_u(t) = [x_u(t), y_u(t), h_u(t)]$, and the coordinates of the end-user as $L_{n_k} = [x_{n_k}, y_{n_k}, 0]$, and the coordinates of the MEC servers are $L_{s_k} = [x_{s_k}, y_{s_k}, 0]$, where $\mathcal{K} \in (1, 2, \ldots, k, \ldots, K)$ denotes the corresponding serial number. Unless otherwise stated, the important notations used in this paper are summarized in Table 2.

**Table 2.** List of Notations.

| Notations | Definitions |
|---|---|
| $\mathcal{N}$ | The set of end-user $n$ |
| $\mathcal{S}$ | The set of MEC server $s$ |
| $\mathcal{U}$ | The set of unmanned aerial vehicle $u$ |
| $\mathcal{T}$ | The set of time slot $t$ |
| $\mathcal{K}$ | The maximum number of end users or MEC servers |
| $L_{n_k}(t)$ | The location of the end-user $n_k$ |
| $L_{s_k}(t)$ | The location of the MEC server $s_k$ |
| $L_u(t)$ | The location of the UAV |
| $g_{n_k,u}(t)$ | The channel gain between the end-user $n_k$ and the UAV $u$ |
| $r_{n_k,u}(t)$ | The transmission rate between the end-user $n_k$ and the UAV $u$ |
| $g_{u,s_k}(t)$ | The channel gain between the UAV $u$ and the MEC server $s_k$ |
| $r_{u,s_k}(t)$ | The transmission rate between the UAV $u$ and the MEC server $s_k$ |
| $t_{fly}(t)$ | The flight delay of UAV $u$ |
| $t_{tr}^{u,n_k}(t)$ | The transmission delay between the end-user $n_k$ and the UAV |
| $t_{cal}^{u}(t)$ | The channel gain between the MEC server $s_k$ and the UAV $u$ |
| $t_{cal}^{n_k}(t)$ | The calculation delay of the end-user $n_k$ |
| $t_{tr}^{u,s}(t)$ | The transmission delay between the UAV and the MEC server $s_k$ |
| $D_{n_k}(t)$ | The computing tasks that end-user $n_k$ needs to complete |
| $R_{uav}(t)$ | The offloading ratio of UAV |
| $R_{s_k}(t)$ | Whether to further offload to the MEC server $s_k$ |
| $S(t)$ | The total size of the discarded tasks in time slot $t$ |

#### 2.1.1. Communication Model

The UAV offers services to all end-users but only serves one end-user in each time slot. We assume that all end-users are fixed at a certain coordinate $L_{n_k} = [x_{n_k}, y_{n_k}, 0]$. At the beginning of the whole decision episode, the UAV $u$ is deployed at the initial position $L_u(0) = [x_u(0), y_u(0), h_u(0)]$. When a certain end-user needs to provide services, the UAV flies directly above the end-user and establishes the communication link. Similar to [18], the communication links are presumed to be dominated by the LOS channels. Thus, the channel gain between end-user $n_k$ and the UAV $u$ could be denoted as

$$g_{n_k,u}(t) = \alpha_0 d_{n_k,u}^{-2}(t) = \frac{\alpha_0}{\left\| L_{n_k} - L_u(t) \right\|^2} \tag{1}$$

where

$$\left\| L_{n_k} - L_u(t) \right\| = \sqrt{\left[ x_{n_k} - x_u(t) \right]^2 + \left[ y_{n_k} - y_u(t) \right]^2 + \left[ 0 - h_u(t) \right]^2},$$

$d_{n_k,u}(t)$ denotes the Euclidean distance between the end-user $n_k$ and the UAV $u$, $\| \cdot \|$ denotes the Euclidean norm, and $\alpha_0$ denotes the received power at the reference distance of 1 m for the transmission power of 1 W. Considering the blocking of the communication signal by the building, the wireless transmission rate can be denoted as

$$r_{n_k,u}(t) = B \log_2 \left( 1 + \frac{P_{down} g_{n_k,u}(t)}{\sigma^2 + f_{n_k,u}(t) P_{NLOS}} \right) \tag{2}$$

where $B$ denotes the assigned communication bandwidth, $P_{down}$ denotes the received power of the UAV, $\sigma^2$ denotes the noise power, $P_{NLOS}$ denotes the transmission loss, $f_{n_k,u}(t)$ denotes whether there is a communication block between end-user $n_k$ and the UAV in time slot $t$ (that is, 0 means no blocking, and 1 means blocking). Similarly, when the UAV needs

further to send the computing tasks to the remote MEC server, the channel gain between the UAV $u$ and the MEC server $s_k$ could be denoted as

$$g_{u,s_k}(t) = \alpha_0 d_{u,s_k}^{-2}(t) = \frac{\alpha_0}{\left\| L_u(t) - L_{s_k} \right\|^2} \tag{3}$$

where

$$\left\| L_u(t) - L_{s_k} \right\| = \sqrt{\left[ x_u(t) - x_{s_k} \right]^2 + \left[ y_u(t) - y_{s_k} \right]^2 + \left[ h_u(t) - 0 \right]^2},$$

Similarly, the wireless transmission rate between the UAV $u$ and MEC server $s_k$ could be denoted as

$$r_{u,s_k}(t) = B \log_2 \left( 1 + \frac{P_{up} g_{u,s_k}(t)}{\sigma^2 + f_{u,s_k}(t) P_{NLOS}} \right) \tag{4}$$

2.1.2. Computation Model

Due to the limited computing resource of the end-user, our proposed offloading decision optimization algorithm is applied to each time slot. According to the offloading policy, the end-user offloads part of the tasks to the UAV, and then the UAV determines to process it locally or further offload to the MEC server. It should be noted that compared with the entire communication and calculation delay, the time to divide the task is very short, so this part of the delay is ignored in our model. In addition, in some computing-intensive applications, such as video analysis, the output data size of the computing results is often much smaller than the input data size. Therefore, the delay of the downlink is also ignored. The key components of the total delay during the offloading process are described as follows.

- The flight delay from the previous location to the end-user directly above;
- The transmission delay from the end-user to the UAV;
- The calculation delay of the UAV;
- The calculation delay of the end-user;
- The transmission delay from the UAV to the MEC server;
- The calculation delay of the MEC server.

The flight delay from the previous location of the UAV $u$ to the end-user directly above could be described as

$$t_{fly}(t) = \frac{\sqrt{\left[ x_{n_k}(t) - x_u(t) \right]^2 + \left[ y_{n_k}(t) - y_u(t) \right]^2}}{v_u} \tag{5}$$

where $v_u$ is the average flight speed of the UAV $u$. The transmission delay from end-user $n_k$ to UAV $u$ could be described as

$$t_{tr}^{u,n_k}(t) = \frac{R_{uav}(t) D_{n_k}(t)}{r_{u,n_k}(t)} \tag{6}$$

where $R_{uav}(t) \in [0, 1]$ is the offloading rate of the end-user $n_k$ to the UAV, and $D_{n_k}(t)$ is the computing task size of the end-user $n_k$ in time slot $t$. The calculation delay of the UAV $u$ could be described as

$$t_{cal}^{u}(t) = \frac{R_{uav}(t) D_{n_k}(t) s}{f_{uav}} \tag{7}$$

where $s$ denotes the CPU cycles required to process each byte, and $f_{uav}$ denotes the calculation frequency of the MEC servers' CPU. Similar to (7), the local calculation delay of the end-user $n_k$ in time slot $t$ could be denoted as

$$t_{cal}^{n_k}(t) = \frac{(1 - R_{uav}(t)) D_{n_k}(t) s}{f_{n_k}} \tag{8}$$

where $f_{n_k}$ denotes the calculation frequency of the end-user $n_k$. According to the offloading policy, it is decided whether to offload the computing tasks to the MEC servers. Due to the limited battery capacity, we consider offloading all the computing tasks received by the UAV to the MEC servers. Therefore, the transmission delay from the UAV $u$ to the MEC server $s_k$ could be denoted as

$$t_{tr}^{u,s_k}(t) = \frac{R_{uav}(t)D_{n_k}(t)}{r_{u,s_k}(t)} \tag{9}$$

The calculate delay of the MEC server $s_k$ could be denoted as

$$t_{cal}^{s_k}(t) = \frac{R_{uav}(t)D_{n_k}(t)s}{f_{s_k}} \tag{10}$$

To define the service delay of each time slot, we assume that the UAVs and the MECs server can only start executing the computing tasks after the transmission is completed to ensure the reliability of the calculation result. We also assume that the end-users execute locally and transmit computing tasks at the same time. Based on the above assumption, the service delay of each time slot could be denoted as

$$T(t) = \begin{cases} t_{cal}^{n_k}(t), & \text{For end-user only.} \\ t_{fly}(t) + t_{tr}^{u,n_k}(t) + t_{cal}^{u}(t), & \text{For end-user and the UAV.} \\ t_{fly}(t) + t_{tr}^{u,n_k}(t) + t_{tr}^{u,s_k}(t) + t_{cal}^{s_k}(t), & \text{For end-user, the UAV, and the MEC server.} \end{cases} \tag{11}$$

#### 2.1.3. Energy Model

Battery capacity has always been a bottleneck in UAV applications. The battery capacity of the UAV is denoted as $E_{\text{battery}}$ . At the beginning of the decision episode, the UAV is in a fully charged state. The UAV continues to serve the end-user until the battery capacity is exhausted. Our study mainly focuses on the calculation and transmission energy consumption of the UAV while ignoring other energy consumption, which has nothing to do with our decision-making. The key components of the energy consumption during the offloading process are described as follows.

- The flight energy consumption of the UAV;
- The transmission energy consumption when UAV receives tasks from end-users;
- The calculation energy consumption of the UAV;
- The transmission energy consumption from the UAV to the MEC server.

The flight energy consumption of the UAV could be denoted as

$$E_{fly}(t) = Pt_{fly}(t) = Fv_u t_{fly}(t) \tag{12}$$

where $F = m_{uav} * g$, which is related to the weight of the UAV. The transmission energy consumption when UAV receives tasks from end-users could be denoted as

$$E_{tr}^{n_k,u}(t) = P_{down} t_{tr}^{u,n_k}(t) \tag{13}$$

where $P_{down}$ denotes the received power of the UAV, and $t_{tr}^{u,n_k}(t)$ denotes the transmission delay. Similar to [28], we model that the calculation power is positively correlated with computing capacity, i.e., $\kappa(f_{uav})^3$, where $\kappa$ denotes the energy consumption factor. The UAV calculation energy consumption is denoted as

$$E_{cal}^{u}(t) = \kappa(f_{uav})^3 t_{cal}^{u}(t) \tag{14}$$

The sending power of the UAV is denoted as $P_{up}$, and the transmission energy consumption of the UAV could be denoted as

$$E_{tr}^{u,s_k}(t) = P_{up} t_{tr}^{u,s_k}(t) \tag{15}$$

According to the above analysis, the total energy consumption of the UAV could be denoted as

$$E_u(t) = \begin{cases} E_{fly}(t) + E_{tr}^{n_k,u}(t) + E_{cal}^{u}(t), & \text{For UAV and end-user.} \\ E_{fly}(t) + E_{tr}^{n_k,u}(t) + E_{tr}^{u,n_k}(t), & \text{For UAV, end-user, and MEC server.} \end{cases} \tag{16}$$

### *2.2. Problem Formulation*

Our study objective is to minimize the weighted total cost of the service delay, energy consumption of the UAV, and the size of the discarded tasks through optimize the offloading policy. The joint optimization problem could be denoted as

$$\min_{L_u(t),R_{uav}(t),R_{mec}(t)} \sum_{t\in\mathcal{T}} \{E_u(t) + \rho_1 T(t) + \rho_2 S(t)\} \tag{17}$$

$$s.t. \quad L_u(t) \in \{(x(t), y(t) | x(t) \in [0, L], y(t) \in [0, W]\}, \forall\, t \in \mathcal{T} \tag{18}$$

$$\sum_{t=1}^{\mathcal{T}} E_u(t) \le E_{battery}, \forall\, t \in \mathcal{T} \tag{19}$$

$$0 < R_{uav}(t) < 1, \forall\, t \in \mathcal{T} \tag{20}$$

$$R_{mec}(t) \in \{0, 1\}, \forall\, t \in \mathcal{T} \tag{21}$$

$$\sum_{t=1}^{\mathcal{T}} \sum_{k=1}^{K} D_{n_k}(t) = D, \forall\, t \in \mathcal{T}, \forall\, k \in K \tag{22}$$

$$0 < S(t) < D_{n_k}(t), \forall\, t \in \mathcal{T} \tag{23}$$

where $\rho_1, \rho_2 > 0$ in (17) are the parameters that define the relative weight, and $S(t)$ denotes the total size of the discarded tasks in time slot $t$. Constraint (18) limits the UAV's range of movement. Constraint (19) means the total energy consumption during the decision episode cannot exceed the maximum battery capacity of the UAV. Constraint (20) denotes the value range of the offloading ratio. Constraint (21) denotes whether to further offload to the MEC server. In (22), $D$ denotes the total size of computing tasks that should be executed during the decision episode. Constraint (23) denotes that the size of the discarded tasks does not exceed the total size of computing tasks in each time slot.

## 3. Soft Actor–Critic Based Dynamic Computation Offloading Algorithm

Our study objective is to obtain the optimal offloading policy by minimizing the weighted total cost of the service delay, energy consumption of the UAV, and the size of the discarded task during the entire decision episode. We consider the standard reinforcement learning framework [29] and formulate the UAV-assisted edge computing offloading decision-making and resource-allocating problem as a Markov decision process (MDP).

### *3.1. Markov Decision Process*

The MDP is usually described as a quintuple $M = \langle S, A, P, R, \gamma \rangle$, which denotes state, action, state transition probability, reward, and discount factor, respectively. The FMC controller used in our proposal can obtain all global information of the end-user, the UAV, and the MEC server. Therefore, the DRL-based algorithm we proposed runs on the FMC controller in each time slot. Furthermore, the state space, action space, and reward are defined as follows.

- State space: We consider the current location of the UAV $L_u(t)$, the UAV battery capacity $E_{battery}(t)$, and the size of computing tasks $D_{n_k}(t)$ as the current state. Therefore, the state space can be denoted as

$$s(t) = \left\{ L_u(t), E_{battery}(t), D_{n_k}(t) \right\} \tag{24}$$

- Action space: We consider the offloading rate $R_{uav}(t)$, whether to further offload to the MEC servers $R_{s_k}(t)$ as the current action of the agent. Therefore, action space can be denoted as

$$\mathrm{a(t)} = \left\{ R_{uav}(t), R_{s_k}(t) \right\} \tag{25}$$

- Reward: We define cumulative rewards to minimize the weighted sum of service delay, energy consumption, and the size of discarded task. Thus, rewards can be denoted as

$$r(t) = -(E_u(t) + \rho_1 * T(t) + \rho_2 * S(t)) \tag{26}$$

where $\rho_1, \rho_2 > 0$ denote the relative weight.

### *3.2. Soft Actor–Critic DRL Algorithm*

Previous studies have shown that DRL algorithms can solve the offloading decision problems in the MEC environment [21,22,30]. However, those DRL algorithms suffer from two main problems: high sample complexity (large amounts of data needed) and other being their brittleness with respect to learning rates, exploration constants, and other hyperparameters. Algorithms, such as DDPG and twin delayed DDPG (TD3), are used to tackle the challenge of high sample complexity in actor–critic frameworks with continuous action spaces. However, they still suffer from brittle stability with respect to their hyperparameters.

Soft actor–critic (SAC) algorithm introduces an actor–critic framework for arrangements with continuous action spaces where in the standard objective of reinforcement learning, i.e., maximizing expected cumulative reward, is augmented with an additional objective of entropy maximization, which provides a substantial improvement in exploration and robustness. Thus, the optimization objective of SAC algorithm is described as

$$\pi^* = \arg\max_{\pi} \sum_{t} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_t))] \tag{27}$$

where $\alpha > 0$ is the temperature parameter, which determines the relative importance of the entropy term against the reward. $\mathcal{H}$ represents the entropy function. The entropy of a random variable $x$ following a probability distribution $P$ is defined as $\mathcal{H}(P) = \mathbb{E}_{x\sim P}[-\log P(x)]$. Similar to the traditional actor–critic algorithm, value function $V^\pi(s)$ and state-value function $Q^\pi(s,a)$ could be defined in the SAC algorithm, which are given as follows

$$V^\pi(s) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t (R(s_t,a_t,s_{t+1}) + \alpha H(\pi(\cdot \mid s_t))) \mid s_0 = s\right] \tag{28}$$

$$Q^\pi(s,a) = \mathop{\mathrm{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t,s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot \mid s_t)) \mid s_0 = s, a_0 = a\right] \tag{29}$$

According to above analysis, $V^\pi$ and $Q^\pi$ are connected by:

$$V^\pi(s) = \mathop{\mathrm{E}}_{a\sim\pi}[Q^\pi(s,a)] + \alpha H(\pi(\cdot \mid s)) \tag{30}$$

and the Bellman equation for $Q^\pi$ is

$$\begin{aligned} Q^\pi(s,a) &= \mathop{\mathrm{E}}_{\substack{s'\sim P \\ a'\sim\pi}}\left[R(s,a,s') + \gamma\left(Q^\pi(s',a') + \alpha H(\pi(\cdot \mid s'))\right)\right] \\ &= \mathop{\mathrm{E}}_{s'\sim P}\left[R(s,a,s') + \gamma V^\pi(s')\right] \end{aligned} \tag{31}$$

SAC learns a policy $\pi_\theta$ and two $Q$ functions $Q_{\phi_1}, Q_{\phi_2}$ and their target networks concurrently. The two Q-functions are learned in a fashion similar to TD3, where a common target is considered for both the Q functions, and clipped double Q-learning is used to train the network. The action-value for one state-action pair can be approximated as

$$Q^\pi(s,a) \approx r + \gamma\left(Q^\pi(s',\tilde{a}') - \alpha \log \pi(\tilde{a}' \mid s')\right), \quad \tilde{a}' \sim \pi(\cdot \mid s') \tag{32}$$

where $\tilde{a}'$ (action taken in next state) is sampled from the policy.

SAC also uses replay buffer like other off-policy algorithms. The quintuple $(s,a,r,s',d)$ from each episode is stored into the replay buffer $\mathcal{D}$. Batches of these transitions are sampled while updating the network parameters.

Just like TD3, SAC uses clipped double Q-learning to calculate the target values for the Q-value network. The target is given by

$$y^t(r,s',d) = r + \gamma\left(\min_{j=1,2} Q_{\phi_{\text{targ}},}(s',\tilde{a}') - \alpha \log \pi_\theta(\tilde{a}' \mid s')\right) \tag{33}$$

where $\tilde{a}'$ is sampled from the policy. The loss function can be defined as

$$L(\phi_i, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d)\sim D}\left[\left(Q_{\phi_i}(s,a) - y^t(r,s',d)\right)^2\right] \tag{34}$$

The main objective of policy optimization will be to maximize the value function, which, in this case, can be defined as

$$V^\pi(s) = \mathbb{E}_{a\sim\pi}[Q^\pi(s,a) - \log \pi(a \mid s)] \tag{35}$$

In SAC, a reparameterization trick is used to sample actions from the policy to ensure that sampling from the policy is a differentiable process. The policy is now parameterized as

$$\tilde{a}'_t = f_\theta(\xi_t; s_t) \tag{36}$$

$$\tilde{a}'_\theta(s,\xi) = \tanh(\mu_\theta(s) + \sigma_\theta(s) \odot \xi) \tag{37}$$

$$\xi \sim \mathcal{N}(0,1) \tag{38}$$

The maximization objective is now defined as

$$\max_\theta \mathbb{E}_{(s\sim D,\xi\sim\mathcal{N})}\left[\min_{j=1,2} Q_{\phi_j}(s,\tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi) \mid s)\right] \tag{39}$$

The pseudocode of soft actor–critic algorithm is given in Algorithm 1 [31]. We optimize the computation offloading policy via the soft actor–critic DRL algorithm in each time slot, thereby minimize the optimization objective. The pseudocode of the SAC-based UAV-assisted computation offloading algorithm is given in Algorithm 2.

**Algorithm 1** Soft actor–critic algorithm

**Input:** $\theta_1, \theta_2, \phi$

$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$
$\mathcal{D} \leftarrow \emptyset$
**for** each iteration **do**
  **for** each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t \mid \mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  **end for**
  **for** each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$
  **end for**
**end for**

**Output:** $\theta_1, \theta_2, \phi$

**Algorithm 2** SAC-based dynamic computation offloading algorithm (SACDCO)

**Input:** The initial location of the UAV $L_u(t)$, the initial battery capacity of the UAV $E_{\text{battery}}(t)$, and task size $D_{n_k}(t)$.

**for** each time slot $t \in \mathcal{T}$ **do**
  Observe state $s(t) = \left\{L_u(t), E_{battery}(t), D_{n_k}(t)\right\}$ and select action $a(t) = \left\{R_{uav}(t), R_{s_k}(t)\right\}$ based on policy parameter $\theta$;
  Select an end-user $n_k$ and generate signal block randomly;
  **if** $R_{uav}(t)! = 0$ **then**
    The UAV flies directly above the end-user $n_k$;
    Calculate the flight distance of the UAV via (5);
    **if** flight distance !=0 **then**
      Calculate flight delay and flight energy consumption via (5) and (11);
    **end if**
    **if** $R_{s_k}(t) == 1$ **then**
      Calculate the channel gain $g_{n_k,u}(t)$ and $g_{u,s_k}(t)$ via (1) and (3);
      Calculate transmission rate $r_{n_k,u}(t)$ and $r_{u,s_k}(t)$ via (2) and (4);
      Calculate the transmission delay $t_{tr}^{u,n_k}(t)$ via (6), the transmission delay $t_{tr}^{u,s_k}(t)$ (9), and the calculation delay $t_{cal}^{s_k}(t)$ via (10), respectively;
      Calculate actual delay via (16);
      Calculate energy consumption via (15);
    **else**
      Calculate transmission delay, calculation delay of the UAV and via (6)–(8), respectively;
      Calculate actual delay via (16);
      Calculate UAV's energy consumption of transmission and calculation via (12) and (13);
      Calculate actual energy consumption via (15);
    **end if**
  **else**
    Calculate local calculation delay via (8);
  **end if**
  Calculate episode reward via (26);
  Update policy parameter $\theta$ via Algorithm 1.
**end for**

## 4. Performance Evaluation

In this section, a detailed numerical evaluation is presented to study the performance of our proposed SACDCO algorithm compared to other baseline schemes. In our proposal, all algorithms and the corresponding simulations are implemented based on Python and executed on a desktop computer with Intel Core i7-8700 6 cores CPU and 32 GB RAM.

### *4.1. Simulation Settings*

As mentioned above, our proposed UAV-assisted MEC system consists of three entities: end-users, UAVs, and edge servers, which is different from previous studies. The offloading approaches in existing literature are not directly applicable to our settings. Thereby, we consider the following intuitive schemes as the baseline schemes.

- (*Local-Only Scheme*) Only execute computing tasks by the end-user;
- (*UAV-Only Scheme*) Only execute computing tasks on the UAV without further offloading to any MEC servers;
- (*Fixed Local-UAV Scheme*) Half of the computing tasks is executed locally while the other half is executed on the UAV.

In our proposal, we consider a two-dimensional square area, in which four end-users are distributed in an area of $L \times W = 500 \times 500$ m$^2$ and fixed positions. Furthermore, four MEC servers are deployed at the edge of the area, and each MEC server is equipped with 8 cores 3.0 GHz CPU. At the beginning of the decision episode, we assume that the UAV is deployed at an initial position $L_u = (250, 250)$ with a height of $H$ = 100 m. For UAV, we refer to the parameters of DJI Air 2S [32]. Unless otherwise stated, the simulation parameters are summarized in Table 3.

**Table 3.** Simulation Parameters.

| Parameters | Values | Parameters | Values |
|---|---|---|---|
| $\alpha_0$ | $-50$ dBm | $L$ | 500 m |
| $B_{n_k}$ | 10 MHz | $W$ | 500 m |
| $B_{uav}$ | 30 MHz | $H$ | 100 m |
| $P_{up}$ | 0.1 w | $L_{n_1}$ | $(125, 125)$ |
| $P_{down}$ | 1 w | $L_{n_2}$ | $(375, 125)$ |
| $\sigma^2$ | $-100$ dBm | $L_{n_3}$ | $(125, 375)$ |
| $P_{NLOS}$ | $-80$ dBm | $L_{n_4}$ | $(375, 375)$ |
| $v_u$ | 15 m/s | $L_{s_1}$ | $(0, 0)$ |
| $D_{n_k}$ | 80 Mbit | $L_{s_2}$ | $(500, 0)$ |
| $s$ | 1000 | $L_{s_3}$ | $(0, 500)$ |
| $m_{uav}$ | 0.6 kg | $L_{s_4}$ | $(500, 500)$ |
| $E_{battery}$ | $1.5 \times 10^5$ J | $f_{n_1}$ | 0.4 GHz |
| $f_{uav}$ | 3.0 GHz $\times$ 2 | $f_{n_2}$ | 0.6 GHz |
| $f_{mec}$ | 3.0 GHz $\times$ 4 | $f_{n_3}$ | 0.8 GHz |
| $g$ | 10 m/s | $f_{n_4}$ | 1.0 GHz |
| $K$ | $1 \times 10^{-28}$ | | |

### *4.2. Simulation Result*

In this section, we have verified the convergence of the SACDCO algorithm and the simulation results of different hyperparameters on the SACDCO algorithm to select the optimal hyperparameters. To prove the importance of offloading policy optimization, we have compared our proposed SACDCO approach with other baseline schemes regarding the delay, energy consumption, and the size of discarded tasks. Subsequently, we have also studied the influence of different UAV parameters on offloading decision-making. The specific simulation results and corresponding charts are given as follows.

Firstly, we study the impact of different learning rates on the cumulative reward of the SACDCO algorithm. As shown in Figure 2, when the learning rate is set to 0.003 and

0.0003, higher cumulative rewards can be obtained. Compared with the learning rate of 0.003, when the learning rate is 0.0003, the cumulative reward of the SACDCO algorithm can converge faster. Therefore, the following experimental results will be based on the learning rate set to 0.0003.

Secondly, we study the impact of different relative weight $\rho_1$, $\rho_2$ on the cumulative reward of the SACDCO algorithm. Considering the value range of the delay, energy consumption, and the size of discarded tasks, we design four schemes and conduct the corresponding simulation. According to the simulation result shown in Figure 3, we find that the SACDCO algorithm could get the highest cumulative reward when $\rho_1$ set to 0.01, and $\rho_2$ set to 0.1. Similarly, the following experimental results will be based on $\rho_1$ set to 0.01 and $\rho_2$ set to 0.1.

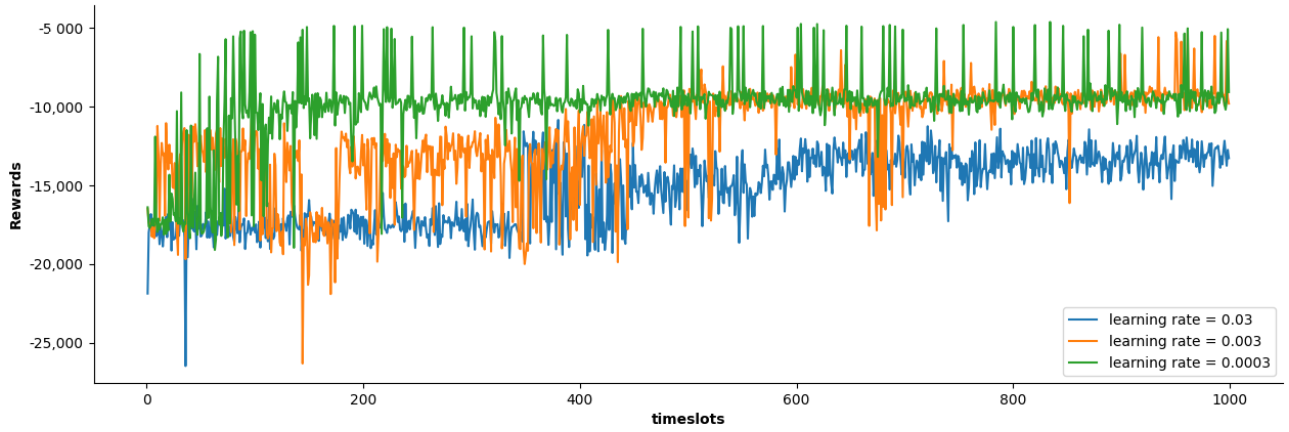


**Figure 2.** Comparison of cumulative rewards of SACDCO algorithm under different learning rates.

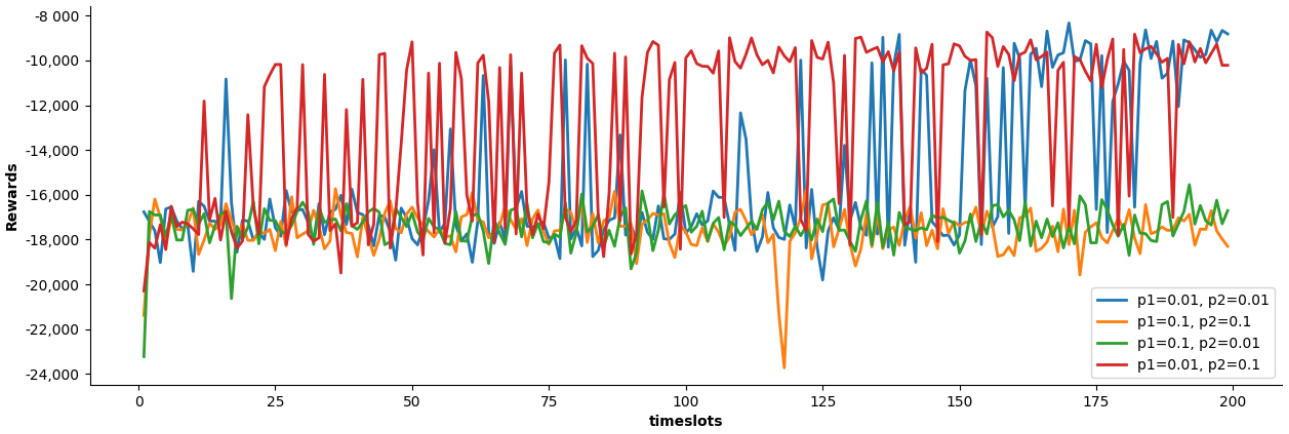


**Figure 3.** Comparison of cumulative rewards of SACDCO algorithm under different relative weight.

To prove the importance of offloading policy optimization, we compared our proposal with other baseline schemes in terms of delay, as shown in Figure 4. Since it is not affected by the local calculation delay, the UAV-only scheme has the lowest delay, about 32 s. After the SACDCO algorithm optimizes the offloading policy, the delay converges to about 40 s. The delays of the other two schemes are relatively high, around 80 s and 90 s, respectively.

The comparison of different schemes in terms of energy consumption of UAV is shown in Figure 5. Since the local-only scheme does not consume the energy of the UAV, the corresponding result does not appear in the figure. We find that the energy consumption generated by our proposed SACDCO approach is at a low level, which is about 2900 KJ (accounted for 60% of the highest energy consumption scheme only).

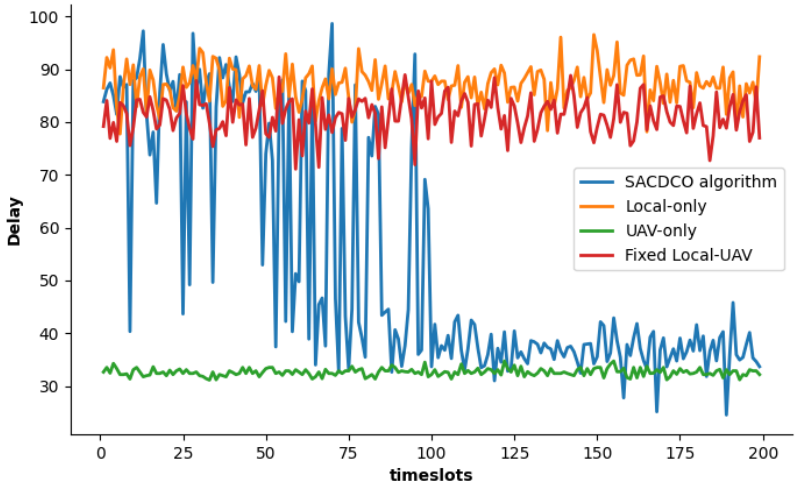


**Figure 4.** Comparison of delay under different schemes.

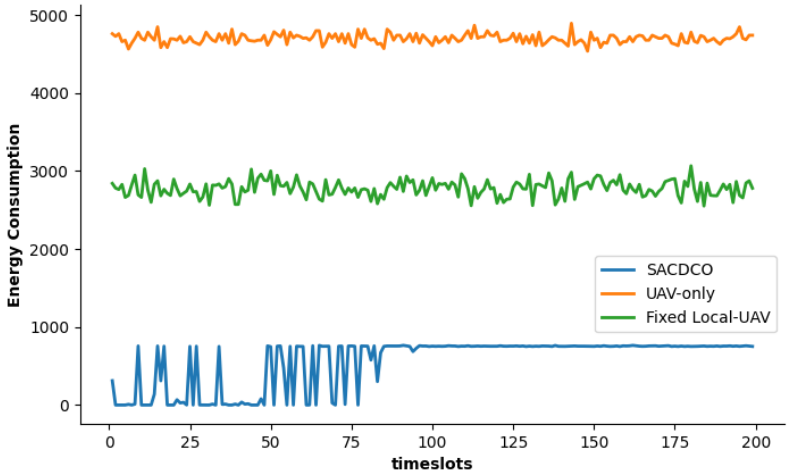


**Figure 5.** Comparison of energy consumption under different schemes.

As shown in Figure 6, we compare our proposal with UAV-only and fixed local-UAV in terms of the size of discarded tasks. When UAV's battery runs out, the remainder of offloaded tasks on the UAV will be discarded. The simulation result shows that the size of the discarded tasks generated by our proposed SACDCO algorithm is at a low level, which is about 35 Mbit (accounted for 43% of the size of total tasks).

We then study the impact of UAV computing capability and bandwidth on the weighted sum of delay, energy consumption, and the size of discarded tasks. We adjust the UAV's computing power by changing the number of UAV's CPU cores, and the simulation result is shown in Figure 7. The corresponding result shows that the proposed SACDCO algorithm could obtain the highest cumulative rewards when the UAV is equipped with two cores. Then, we study how cumulative rewards behave as the UAV bandwidth increase from 5GHz to 30GHz. We observe that as the bandwidth increases, the cumulative reward of the SACDCO algorithm will also increase. Compared with other baseline algorithms, the SACDCO algorithm can maintain good performance. It should be noted that too high bandwidth is usually impractical. The simulation result is given in Figure 8.

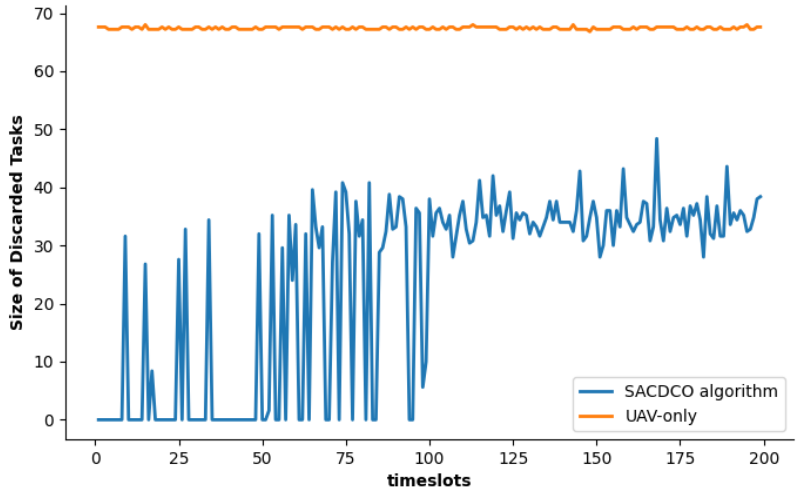


**Figure 6.** Comparison of the size of discarded tasks under different schemes.

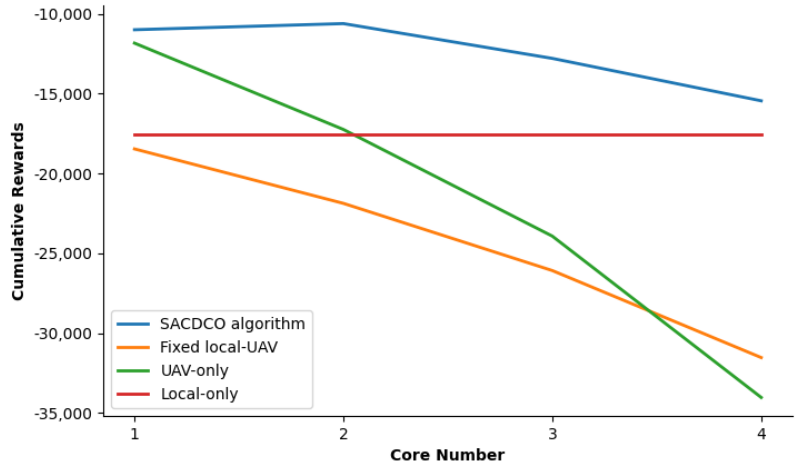


**Figure 7.** Comparison of cumulative rewards for different UAV computing capabilities under different schemes.

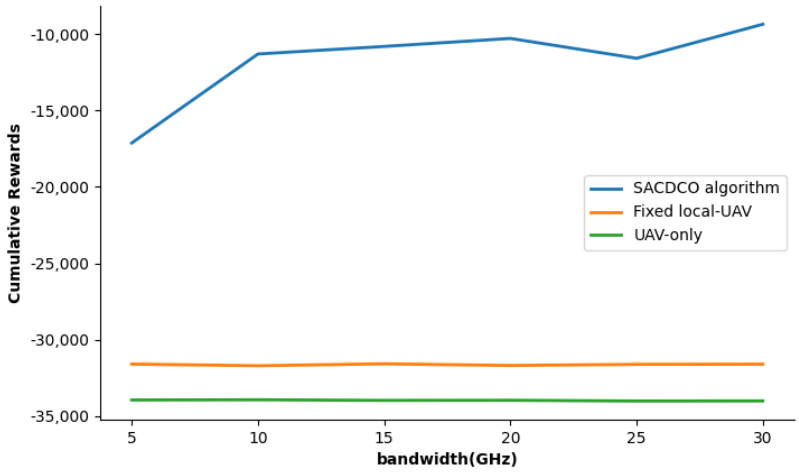


**Figure 8.** Comparison of cumulative rewards for different UAV bandwidth under different schemes.

## 5. Conclusions

Considering stochastic computation tasks generated by end-users, the mobility of the UAV, and the limited battery capacity of the UAV, we have studied the computation offloading decision problem in the UAV-assisted MEC environment with multiple users

and multiple MEC servers. To obtain the global optimal offloading policy, we minimize the weighted total cost of system delay, energy consumption, and the size of discarded tasks as the optimization objective. We propose the soft actor–critic dynamic computation offloading approach to optimize computation offloading and resource allocation policy. Unlike previous studies, we consider letting UAV and MEC servers work collaboratively to provide computing services for end-users. To this end, we consider three intuitive schemes as the baseline schemes in the simulation, i.e., local-only scheme, UAV-only scheme, and fixed local-UAV scheme, respectively. Extensive simulations have demonstrated the superiority of our proposal in terms of delay, energy consumption, and the size of discarded tasks. In particular, compared with the fixed local-UAV scheme in the specific simulation setting, our proposed approach reduces system delay and energy consumption by approximately 50% and 200%, respectively.

In the future, we will extend the research on offloading decision optimization to a multi-UAV collaborative edge computing scenario. Existing literature has shown the superiority of multi-UAV collaborative assisted edge computing. In [23], multiple UAVs work collaboratively to service IoT devices, and a DQN-based optimization approach was proposed to improve the efficiency of each UAV while maintaining the QoS of ground IoT devices. Nevertheless, the DQN algorithm could not handle high-dimensional action problems well, as we mentioned before. Savkin et al. [33] proposed a multi-UAV collaborative approach for improving the network performance between the UAV and the base stations. Chen [34] proposed an approach for improving pairing rate through optimizing the power allocations, UAVs' locations, and nodes scheduling. However, their optimization objective only focuses on system delay, which is different from ours. In the future, we will introduce multi-agent reinforcement learning algorithms to our proposal, which could further improve the algorithm's performance. We hold the opinion that multi-agent reinforcement learning algorithms will play an indispensable role in complex decision problems.

**Author Contributions:** Conceptualization, S.L.; methodology, S.L.; software, S.L.; validation, S.L.; formal analysis, S.L.; investigation, S.L.; resources, S.L.; data curation, S.L.; writing—original draft preparation, S.L.; writing—review and editing, S.L., X.H., and Y.D.; visualization, S.L.; supervision, X.H. and Y.D.; project administration, X.H. and Y.D.; funding acquisition, X.H. and Y.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was partially supported by the National Natural Science Foundation of China (11461038) and the Innovation Foundation of Colleges and Universities in Gansu Province (2020A-033).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Lin, H.; Zeadally, S.; Chen, Z.; Labiod, H.; Wang, L. A survey on computation offloading modeling for edge computing. *J. Netw. Comput. Appl.* **2020**, *169*, 102781. [CrossRef]
2. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. Delay-tolerant sequential decision making for task offloading in mobile edge computing environments. *Information* **2019**, *10*, 312. [CrossRef]
3. Zhang, K.; Mao, Y.; Leng, S.; Maharjan, S.; Zhang, Y. Optimal delay constrained offloading for vehicular edge computing networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
4. Wu, Y.; Qian, L.P.; Ni, K.; Zhang, C.; Shen, X. Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading. *IEEE J. Sel. Top. Signal Process.* **2019**, *13*, 392–407. [CrossRef]
5. You, C.; Zeng, Y.; Zhang, R.; Huang, K. Asynchronous mobile-edge computation offloading: Energy-efficient resource management. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 7590–7605. [CrossRef]
6. Pan, Y.; Chen, M.; Yang, Z.; Huang, N.; Shikh-Bahaei, M. Energy-efficient NOMA-based mobile edge computing offloading. *IEEE Commun. Lett.* **2018**, *23*, 310–313. [CrossRef]

7. Xu, X.; Li, Y.; Huang, T.; Xue, Y.; Peng, K.; Qi, L.; Dou, W. An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. *J. Netw. Comput. Appl.* **2019**, *133*, 75–85. [CrossRef]
8. Zhang, G.; Zhang, W.; Cao, Y.; Li, D.; Wang, L. Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4642–4655. [CrossRef]
9. Zhang, W.; Zhang, Z.; Zeadally, S.; Chao, H.C.; Leung, V.C. MASM: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence. *IEEE Trans. Ind. Inform.* **2019**, *15*, 4216–4224. [CrossRef]
10. Vu, T.T.; Van Huynh, N.; Hoang, D.T.; Nguyen, D.N.; Dutkiewicz, E. Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
11. Neely, M.J. Intelligent packet dropping for optimal energy-delay tradeoffs in wireless downlinks. *IEEE Trans. Autom. Control* **2009**, *54*, 565–579. [CrossRef]
12. Yu, H.; Neely, M.J. A new backpressure algorithm for joint rate control and routing with vanishing utility optimality gaps and finite queue lengths. *IEEE/ACM Trans. Netw.* **2018**, *26*, 1605–1618. [CrossRef]
13. Sharma, G.; Mazumdar, R.; Shroff, N. Delay and capacity trade-offs in mobile ad hoc networks: A global perspective. *IEEE/ACM Trans. Netw.* **2007**, *15*, 981–992. [CrossRef]
14. Mao, Z.; Koksal, C.E.; Shroff, N.B. Near optimal power and rate control of multi-hop sensor networks with energy replenishment: Basic limitations with finite energy and data storage. *IEEE Trans. Autom. Control* **2011**, *57*, 815–829.
15. Zeng, Y.; Zhang, R.; Lim, T.J. Throughput maximization for UAV-enabled mobile relaying systems. *IEEE Trans. Commun.* **2016**, *64*, 4983–4996. [CrossRef]
16. Wu, Q.; Zeng, Y.; Zhang, R. Joint trajectory and communication design for multi-UAV enabled wireless networks. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 2109–2121. [CrossRef]
17. Hu, Q.; Cai, Y.; Yu, G.; Qin, Z.; Zhao, M.; Li, G.Y. Joint offloading and trajectory design for UAV-enabled mobile edge computing systems. *IEEE Internet Things J.* **2018**, *6*, 1879–1892. [CrossRef]
18. Jeong, S.; Simeone, O.; Kang, J. Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning. *IEEE Trans. Veh. Technol.* **2017**, *67*, 2049–2063. [CrossRef]
19. Zhou, F.; Wu, Y.; Hu, R.Q.; Qian, Y. Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1927–1941. [CrossRef]
20. Xiong, J.; Guo, H.; Liu, J. Task offloading in UAV-aided edge computing: Bit allocation and trajectory optimization. *IEEE Commun. Lett.* **2019**, *23*, 538–541. [CrossRef]
21. Shakarami, A.; Ghobaei-Arani, M.; Shahidinejad, A. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. *Comput. Netw.* **2020**, *182*, 107496. [CrossRef]
22. Huang, L.; Feng, X.; Zhang, C.; Qian, L.; Wu, Y. Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. *Digit. Commun. Netw.* **2019**, *5*, 10–17. [CrossRef]
23. Yang, L.; Yao, H.; Wang, J.; Jiang, C.; Benslimane, A.; Liu, Y. Multi-UAV-enabled load-balance mobile-edge computing for IoT networks. *IEEE Internet Things J.* **2020**, *7*, 6898–6908. [CrossRef]
24. Liu, Y.; Xie, S.; Zhang, Y. Cooperative offloading and resource management for UAV-enabled mobile edge computing in power IoT system. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12229–12239. [CrossRef]
25. Zhu, S.; Gui, L.; Zhao, D.; Cheng, N.; Zhang, Q.; Lang, X. Learning-based computation offloading approaches in UAVs-assisted edge computing. *IEEE Trans. Veh. Technol.* **2021**, *70*, 928–944. [CrossRef]
26. Asheralieva, A.; Niyato, D. Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers. *IEEE Internet Things J.* **2019**, *6*, 8753–8769. [CrossRef]
27. Taleb, T.; Ksentini, A. Follow me cloud: Interworking federated clouds and distributed mobile networks. *IEEE Netw.* **2013**, *27*, 12–19. [CrossRef]
28. Wang, Y.; Sheng, M.; Wang, X.; Wang, L.; Li, J. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Trans. Commun.* **2016**, *64*, 4268–4282. [CrossRef]
29. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, UK, 1998; Volume 135.
30. Mohammed, A.; Nahom, H.; Tewodros, A.; Habtamu, Y.; Hayelom, G. Deep reinforcement learning for computation offloading and resource allocation in blockchain-based multi-UAV-enabled mobile edge computing. In Proceedings of the 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 18–20 December 2020; pp. 295–299.
31. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
32. Shenzhen DJI Innovation Technology Co., Ltd. Technical parameters of DJI Air 2S. Available online: https://www.dji.com/air-2s/specs (accessed on 27 August 2021).
33. Savkin, A.V.; Huang, H. Deployment of unmanned aerial vehicle base stations for optimal quality of coverage. *IEEE Wirel. Commun. Lett.* **2018**, *8*, 321–324. [CrossRef]
34. Chen, Q. Joint position and resource optimization for multi-UAV-aided relaying systems. *IEEE Access* **2020**, *8*, 10403–10415. [CrossRef]