Utility-Oriented Optimization for Video Streaming in UAV-Aided MEC Network: A DRL Approach

Jiansong Miao, Shanling Bai, Shahid Mumtaz, *Senior Member, IEEE,*
Qian Zhang, and Junsheng Mu, *Member, IEEE*

Abstract

The integration of unmanned aerial vehicles (UAVs) in future communication networks has received great attention, and it plays an essential role in many applications, such as military reconnaissance, fire monitoring, etc. In this paper, we consider a UAV-aided video transmission system based on mobile edge computing (MEC). Considering the short latency requirements, the UAV acts as a MEC server to transcode the videos and as a relay to forward the transcoded videos to the ground base station. Subject to constraints on discrete variables and short latency, we aim to maximize the cumulative utility by jointly optimizing the power allocation, video transcoding policy, computational resources allocation, and UAV

This work was supported by the Beijing Natural Science Foundation (No.XXXXXXX). (*Corresponding author: Jiansong Miao.*)

Jiansong Miao, Shanling Bai, Qian Zhang, and Junsheng Mu are with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: miaojiansong@bupt.edu.cn; bsl_2021@bupt.edu.cn; qianzhang@bupt.edu.cn; mujs@bupt.edu.cn).

Shahid Mumtaz is with the Department of Applied Informatics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland, and also with the Department of Engineering, Nottingham Trent University, NG1 4FQ Nottingham, U.K. (e-mail: dr.shahid.mumtaz@ieee.org).

flight trajectory. The above non-convex optimization problem is modeled as a Markov decision process (MDP) and solved by a deep deterministic policy gradient (DDPG) algorithm to realize continuous action control by policy iteration. Simulation results show that the DDPG algorithm performs better than deep Q-learning network algorithm (DQN) and actor-critic (AC) algorithm.

Index Terms

UAV, mobile edge computing, video transcoding, utility, DDPG.

I. Introduction

Recently, many forest fires have created new challenges in monitoring the fire conditions. Dynamically monitoring real-time panoramic images of the affected area is essential for fire-fighters [1]. Unmanned aerial vehicles (UAVs) can be a promising solution owing to their unique advantages, such as low cost, high mobility, and the ability to capture high-definition videos [2].

In general, the affected area in deep forest is far from available communication facilities. It is vital to have effective and stable communication with the ground base station. Fortunately, many excellent works (e.g., [3], [4]) have been conducted on the UAV-aided relaying system, providing quick and on-demand wireless connections for the area without network coverage. In addition, a dynamic transcoding policy is considered to accommodate mobile users' dynamic nature better. Although it is a task that consumes large amounts of computational resources, mobile edge computing (MEC) is regarded as a promising scheme [5]. It is a great attempt to leverage the technology of the UAV in MEC networks, which has been shown to be substantial [6]–[9]. MEC makes more personalized video transcoding services for users with low latency.

Although UAVs have several advantages in wireless communication, the lack of energy has become a significant bottleneck limiting the further development of UAV-aided communication systems [7]. Therefore, it is meaningful to improve the energy efficiency. Considering the video quality, the system's utility [10], [11] is established as the main variable to be discussed.

Many works (e.g., [12]–[14]) have used traditional techniques to deal with the problem, which requires acquiring environmental information such as channel conditions and other equipment locations before the UAV takes off [15]. However, the actual deployment environment of the UAV is complex and variable, and many environmental variables cannot be acquired in advance. Therefore, it is practically significant to make immediate decisions while interacting with the environment. In recent years, the rapid development of deep learning has brought a breakthrough in reinforcement learning (RL), and their combination, i.e., deep reinforcement learning (DRL), has overcome the limitations of complex environments and significantly accelerated the training speed of agents.

To address these challenges, we investigate a UAV-aided MEC network for video streaming. It has robust research significance and application value in achieving maximum system's utility by jointly optimizing the power allocation, video transcoding policy, computational resources allocation, and UAV flight trajectory. The main contributions of our work are as follows:

- We propose a UAV-aided MEC network for video streaming. The observation UAVs (OUAVs) capture real-time panoramic images of the monitoring area with a fixed trajectory. A relay UAV (RUAV) is deployed to relay and transcode the videos according to a dynamic transcoding policy. The dynamic selection of transcoding policy is a computationally intensive task. Subject to constraints on short latency, we jointly optimize the power allocation of all UAVs, transcoding policy, computational resources allocation, and flight trajectory of the RUAV to maximize the system's utility.
- We formulate the non-convex optimization with different practical constraints into a Markov decision process (MDP). The approach identifies the distances as states; power allocation, computational resources allocation, moving directions and transcoding policy as actions; and the system's utility as the reward. Faced with the continuity of action space, we apply the deep deterministic policy gradient (DDPG) algorithm to learn the near-optimal policy.
- A Python-based simulator is developed in the simulation to implement the proposed al-

gorithm. In addition, we simulate the deep Q-learning network algorithm (DQN) and the actor-critic (AC) algorithm for comparison. The results show that the proposed DDPG algorithm can achieve higher cumulative utility.

The rest of this paper is arranged as follows. Section II provides the related work. Section III introduces the UAV-aided video transmission system based on MEC and formulates the problem. MDP and the DDPG-based framework are presented in Section IV. In Section V, we present the simulation results and discuss them to demonstrate the effectiveness and superiority of proposed the DDPG algorithm. Finally, we conclude this paper in Section VI.

II. Related Work

A. *Energy Efficiency in UAV-Aided Communication Systems*

The disadvantages of limited batteries accompany the widespread presence of UAV-aided communication systems, and enhancing energy efficiency has become a hot topic. In UAV-aided communication systems, the energy consumption of UAVs is mainly composed of communication energy consumption for signal processing and flight energy consumption to overcome gravity and air resistance [12]. For the communication energy consumption, we can adapt the energy consumption model [16] that has been extensively validated in ground communication. Another fortunate thing is that the flight power of the rotary-wing UAV was modeled as the sum of constant blade profile power, induced power, and parasite power through a rigorous mathematical derivation [12], thereby providing an excellent basis for the study of UAV energy efficiency.

Recently, there have been many works (e.g., [13], [14], [17]) focusing on the energy efficiency issue in UAV-aided communication systems. Wang *et al.* presented a UAV-aided data collection sensor network in [13]. The energy consumption was minimized by path planning and power allocation for data uploading. Constrained by communication power and fuel consumption, Zhang *et al.* maximized the energy efficiency through power control and UAV trajectory design [14].

Conversely, the variables in the communication system are optimized by traditional optimization algorithms, which require the acquisition of environmental information before the departure of the UAV. As for the UAV communication system, various studies have applied the DRL approaches to solve the joint trajectory optimization and resource allocation problems [11], [17]–[19]. With DQN algorithm, Zhan *et al.* in [17] aimed to reduce the energy consumption by jointly optimized trajectory and task completion time allocation. Nasr-Azadani *et al.* in [18] adopted AC algorithm to jointly optimize the user association and UAVs' trajectories. In [11] and [19], efficient trajectory designs have been proposed by employing the multi-agent algorithm. To summarize, most of these existing works try to maximize energy consumption or energy efficiency of UAV-aided systems, nevertheless, these methods cannot be directly used in dynamic adaptive streaming transcoding scenarios.

B. UAV-aided Video Streaming

Since UAVs can capture video streaming from a bird's-eye view via their equipped high definition camera, they play an even more critical role in video streaming applications such as military reconnaissance, natural disaster monitoring, etc. [20]. Nowadays, UAV-aided video streaming has caught great attention in academia. A UAV-aided video streaming system was studied by Zhan *at al.* [21], in which the transmit power, bandwidth, and flight path were jointly optimized to maximize energy efficiency. Medeiros *et al.* in [22] proposed an energy-aware swarm-based and mobility prediction scheme for UAV video delivery. Simulation results showed a high quality of experience (QoE) while minimizing energy consumption.

In video streaming applications, video transcoding, which processes the adaptive bitrates of a video and provides the adaptive video streaming to users, is essential. With MEC, Liu *et al.* proposed a joint approach to improve the transcoding performance while satisfying the resource and latency constraints [23]. Li *et al.* [24] showed high energy efficiency in video streaming with a joint caching and transcoding policy compared to another two policies without transcoding.

The above papers demonstrate that transcoding services at the edge nodes effectively reduce video service latency and improve QoE. However, to our best knowledge, very little attention has been paid to video transcoding in UAV-aided communication systems. Meanwhile, transcoding policies they consider all statically transcode the original videos into several fixed levels, ignoring the potential mismatch between the actual transmission rate of users and bit rate sets of transcoded videos. The low latency of the video is also an aspect that must be taken into account.

C. UAV-Aided MEC Networks

In order for users to better match their transmission rate with low latency, transcoding service is considered for UAV-aided video streaming. Although it is a computationally intensive task, the maturity of UAV-aided MEC technology provides us with a promising solution. Liu *et al.* in [6] took DRL as the tool to minimize the delay for both communication and computation in a maritime UAV swarm MEC network. In [7], Zhao *et al.* cooperatively optimized the trajectories and computational task allocation to reduce execution latency and energy consumption.

Different from [6], [7], the UAVs are used not only as MEC servers but also as relays in [8], [9]; thus, the performance of the network can be further improved by dynamically adjusting the locations. Xu *et al.* investigated a UAV-aided relaying and MEC network [8], and the task completion time was minimized by addressing joint optimization issues. To address the energy consumption minimization problem, Liu *et al.* proposed a three-stage alternative algorithm in [9].

In general, existing researches on UAV-aided video streaming (e.g., [21], [22]) do not consider video transcoding. However, dynamic video transcoding is very valuable for high QoE, and it is not practical to avoid discussing MEC systems [5]. In addition, few researches on UAV-aided MEC networks investigate the system's utility compared to energy consumption and latency (e.g., [6]–[9]). Meanwhile, UAV-aided relaying and MEC networks to enable emergency services need to be studied. Therefore, we attempt to investigate a video transmission scenario based on MEC,

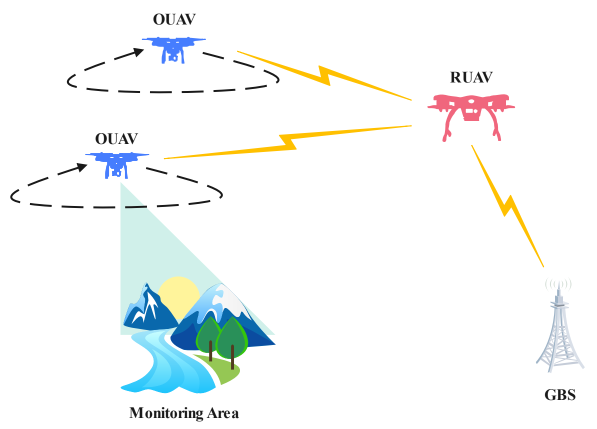


Fig. 1. System model for the UAV-aided MEC network.

where the UAV's video transcoding task is performed to maximize the system's utility. The non-convex problem is modeled as an MDP and solved using a DRL approach. Finally, we obtain the system's cumulative utility.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, a system overview is provided first. Then, we present the communication, video transcoding, and energy consumption models. Finally, we formulate the optimization problem. For clarity, the frequently used notations related to the system model are summarized in Table I.

A. System Overview

The UAV-aided video transmission system based on MEC is shown in Fig. 1. It is composed of a ground base station (GBS) and multiple UAVs. We aim to complete video collection in dangerous areas and complete communication with the GBS through RUAV. The OUAVs are deployed as a monitoring node, shooting videos with a fixed trajectory, and transmitting the video to the RUAV. The RUAV receives the video and transcodes it to realize smooth playback of the video under different channel conditions. When video transcoding is completed, the RUAV forward the transcoded video to the GBS. We assume that such communication scenarios may

TABLE I

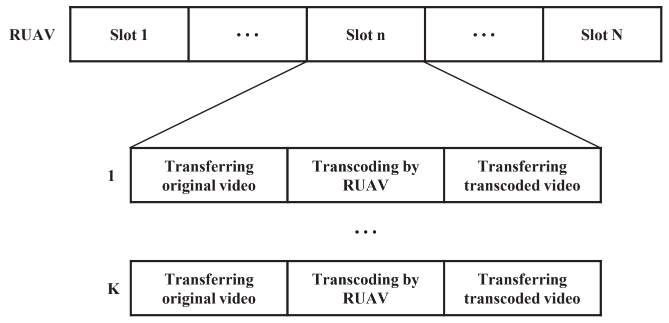
SUMMARY OF KEY NOTATIONS RELATED TO SYSTEM MODEL.

Notations	Description	Notations	Description
δ	the elemental slot length, $T = N\delta$	QoE_k	the user quality-of-experience
$d_{O_k 2R}[n]$	the distance between RUAV and OUAV	$d_{R2B}[n]$	the distance between RUAV and GBS
$h_{O_k 2R}[n]$	channel gain of OUAV-RUAV link	$h_{R2B}[n]$	channel gain of RUAV-GBS link
ω_0	the bandwidth of UAV, $\omega_0 = \frac{\omega}{K+1}$	$r_{O_k 2R}[n]$	the offloading rate of OUAV-RUAV
$r_{R2B}[n]$	the transmission rate of RUAV-GBS	$t_{O,k}$	the offloading time
E_O	the energy consumption to offload	$b_k[n]$	the CPU cycle required to transcode
$a_k[n]$	computational resources radio	$t_{c,k}$	the transcoding time
E_{comp}	the energy consumption to transcode	$t_{r,k}$	the forwarding time
E_R	the energy consumption to forward	$P_F[n]$	the propulsion power consumption
E_F	the propulsion energy consumption	t_D	the system latency
$p_k[n]$	the transmit power of k_{th} OUAV	$p_R[n]$	the transmit power of RUAV

occur in forest fires, environmental protection monitoring, or battlefields where the real-time footage of hard-to-reach core areas is needed for further work. However, in the close range of these scenarios, the necessary communication facilities are difficult to establish or maintain, and the communication quality may be not guaranteed or it may even be impossible to form an effective communication link.

The time slot structure of the communication system is shown in Fig. 2. The OUAVs follow a fixed circular trajectory, and the flight period T is divided into N equal time slots, i.e., $T = N\delta$, where δ denotes the elemental slot length. Since the slot is small enough, the position of the UAVs can be considered unchanged, and each slot is further divided into three sub-slots. The tasks in each sub-slot are as follows: 1) Video transmission between the OUAVs and the RUAV. 2) Video transcoding by the RUAV. 3) Video transmission between the RUAV and the GBS.

The set of OUAVs is denoted as $\mathcal{K}$. In general, a 3D Cartesian coordinate system is utilized in our model. To simplify the problem, the trajectory of the UAVs over T can be linearly



RUAV
Slot 1
···
Slot n
···
Slot N
1
Transferring original video
Transcoding by RUAV
Transferring transcoded video
···
K
Transferring original video
Transcoding by RUAV
Transferring transcoded video

Fig. 2. The time slot structure of the communication system.

discretized into N positions. The coordinates of the k^{th} OUAV and RUAV are denoted by $\boldsymbol{q}_{O_k}[n] = [X_k[n], Y_k[n], H_k]$, where $k \in \mathcal{K} = \{1, 2, \ldots, K\}$, $n \in \mathcal{N} = \{1, 2, \ldots, N\}$ and $\boldsymbol{q}_R[n] = [X_R[n], Y_R[n], H_R]$. The flight height of all UAVs is constant and the position of the GBS is $\boldsymbol{q}_B = [X_B, Y_B, 0]$, so the distance between the k^{th} OUAV and the RUAV is

$$d_{O_k 2R}[n] = \sqrt{(X_R[n] - X_k[n])^2 + (Y_R[n] - Y_k[n])^2 + (H_R - H_k)^2}. \tag{1}$$

The distance between the RUAV and the GBS is

$$d_{R2B}[n] = \sqrt{(X_R[n] - X_B)^2 + (Y_R[n] - Y_B)^2 + H_R^2}. \tag{2}$$

B. Communication Model

As shown in Fig. 1, we consider a computational offloading issue for each OUAV, i.e., the k^{th} OUAV transmits the original video to the RUAV. With few obstacles, the OUAV-to-RUAV channel is usually characterized by line-of-sight (LoS) link, the free-space path loss model [25], [26] is adopted. The channel gain of the OUAV-to-RUAV channel can be expressed as

$$h_{O_k 2R}[n] = \beta_0 d_{O_k 2R}^{-2}[n] = \frac{\beta_0}{(X_R[n] - X_k[n])^2 + (Y_R[n] - Y_k[n])^2 + (H_R - H_k)^2}, \tag{3}$$

where β_0 denotes the channel power at reference distance $d_0 = 1m$.

For the RUAV-to-GBS channel, the buildings, trees, and other structures can bring excessive path loss on top of the free space path loss. The path loss of which is averaged over this case, i.e., line-of-sight (LoS) and non-line-of-sight (NLoS) channels, and given in a probabilistic manner [27]–[29], which can be expressed as

$$PL_{R2B}[n] = 20\log\left(\frac{4\pi f_c d_{R2B}[n]}{v_c}\right) + \eta_{LoS}P_{LoS} + \eta_{NLoS}P_{NLoS}, \tag{4}$$

where f_c denotes the carrier frequency, v_c is the speed of light. P_{LoS} and $P_{NLoS} = 1 - P_{LoS}$ are the probability of LoS and NLoS, respectively. According to [29], P_{LoS} is written as a Sigmoid function of the elevation angle θ,

$$P_{LoS}(\theta) = \frac{1}{1 + a\exp\left[-b\left(\theta - a\right)\right]}, \tag{5}$$

where a and b are S-curve parameters that depend on the specific environment (e.g., urban and rural). The value of θ is calculated by

$$\theta = 180\pi^{-1}\arctan\left(\frac{H_R}{\sqrt{\left(X_R[n] - X_B\right)^2 + \left(Y_R[n] - Y_B\right)^2}}\right). \tag{6}$$

At last, η_{LoS} and η_{NLoS} in (4) are the additional mean losses of LoS and NLoS links, respectively.

Assuming that all the OUAVs operate in an orthogonal frequency division multiple access (OFDMA) mode, there is no interference from OUAVs to the GBS. The available bandwidth ω of the system is divided equally into $K+1$ shares (K OUAVs and 1 RUAV). Then, the available bandwidth of a UAV is $\omega_0 = \frac{\omega}{K+1}$. The transmit power of the k^{th} OUAV for video transmission to the RUAV is $p_k[n]$. Then, the offloading rate of the OUAVs to the RUAV can be expressed as

$$r_{O_k2R}[n] = \omega_0\log_2\left(1 + \frac{p_k[n]h_{O_k2R}[n]}{N_0}\right), k \in \mathcal{K}, \tag{7}$$

where N_0 is the natural noise at the RUAV. The video transmission rate from the RUAV to the GBS is given by

$$r_{R2B}[n] = \omega_0\log_2\left(1 + \frac{p_R[n]}{PL_{R2B}[n]N_0}\right), \tag{8}$$

where $p_R[n]$ is the transmit power of the RUAV for video transmission to the GBS.

C. Video Transcoding Model

Video transcoding is a computationally intensive task that demands many CPU resources to realize the conversion between the source and target bit rates of video [5]. Due to the complex and variable channel conditions, the RUAV chooses the appropriate transcoding rate according to the channel conditions to avoid the excessive transcoding time, which causes video playback lag. Assume that the original video p captured by the k^{th} OUAV can be converted from the original code rate to M levels, i.e., $c = \{c_1, c_2, \ldots, c_M\}$. Typical video captured by OUAV has a uniform original bit rate of 2.75 Mbps and is transcoded by the RUAV's dynamic selection policy to a bit rate of c_k. According to [30], the CPU computation cycle required to transcode the video captured by the k^{th} OUAV at time slot n is given by

$$b_k[n] = \xi {c_k}^{\varsigma}[n], \tag{9}$$

where ξ and ς are hardware-related constants.

For the RUAV, its computational resources are limited, and it is assumed that the maximum computational resource is $f_{\max}$. The proportion of computational resources allocated by the system for each OUAV is $a_k[n]$; then, the time consumed by transcoding can be written as

$$t_{c,k} = \frac{b_k[n]}{a_k[n] f_{\max}}. \tag{10}$$

where $\sum_{k=1}^{K} a_k[n] = 1$, indicating that all videos transmitted per time slot are comparable to the maximum computational capacity of the RUAV.

D. Energy Consumption Model

The energy consumption of the system in this paper is composed of four parts, which are the energy consumption of each OUAV to offload the original video to the RUAV, the energy consumption of the RUAV to transcode video, the energy consumption of the RUAV to forward video to GBS, and the propulsion energy consumption of the OUAVs and RUAV.

1) The size of original video captured by the k^{th} OUAV is $D_k[n]$, therefore, the time consumed by offloading is written as

$$t_{O,k} = \frac{D_k[n]}{r_{O_k 2R}[n]}. \tag{11}$$

Hence, the energy consumption of the OUAVs to offload the original video to the RUAV is written as $E_O = \sum_{k=1}^{K} p_k[n] t_{O,k}$.

2) The time consumed by transcoding has beeen given by (10), hence, the energy consumption of the RUAV to transcode video [31] can be expressed as

$$E_{comp} = \sum_{k=1}^{K} j_R \left(a_k[n] f_{\max}\right)^{s_R} t_{c,k}, \tag{12}$$

where j_R and s_R are the energy consumption coefficients associated with the CPU of the RUAV at unit bits.

3) Assuming that the size of the video after transcoding is $D_k'[n]$, the time required for the RUAV to forward this video to the GBS is

$$t_{r,k} = \frac{D_k'[n]}{r_{R2B}[n]}. \tag{13}$$

Hence, the energy consumption of RUAV to forward video to GBS is $E_R = \sum_{k=1}^{K} p_R[n] t_{r,k}$.

4) The rotary-wing UAV needs a certain amount of propulsion to maintain its flight state, and the propulsion power of the RUAV can be written as [12]

$$P_F[n] = \underbrace{P_{blade}\left(1 + \frac{3v^2[n]}{U_{tip}^2}\right)}_{\text{blade profile}} + \underbrace{P_{induced}\left(\sqrt{1 + \frac{v^4[n]}{4v_0^4}} - \frac{v^2[n]}{2v_0^2}\right)^{\frac{1}{2}}}_{\text{induced}} + \underbrace{\frac{1}{2} d_0 \rho s A v^3[n]}_{\text{parasite}}, \tag{14}$$

where P_{blade} and $P_{induced}$ denote the blade profile power and induced power; U_{tip} and v_0 are the tip speed of the rotor blade and mean rotor induced velocity; d_0 and ρ are the fuselage drag ratio and air density; s and A are the rotor solidity and rotor disc area. $v[n]$ is the average speed of the UAVs, calculated as the *Euclidean* distance of the position,

$$v[n] = \frac{\|\boldsymbol{q}_R[n] - \boldsymbol{q}_R[n-1]\|}{\delta}. \tag{15}$$

Since the OUAVs move according to fixed trajectories and speeds, the propulsion energy consumption of OUAVs is constant. Hence, merely the propulsion energy consumption of RUAV is optimized, which can be expressed as $E_F = P_F[n]\delta$.

Considering that the video captured by different OUAVs can be offloaded, transcoded and forwarded simultaneously, the system latency is the maximum value of the total latency in offloading, transcoding and forwarding, which is given by

$$t_D = \max_k (t_{O,k} + t_{c,k} + t_{r,k}). \tag{16}$$

E. Problem Formulation

We aim to maximize the system's utility with low latency by jointly optimizing the power allocation of the OUAVs and the RUAV, the video transcoding policy, the computational resources allocation, and the flight trajectory of the RUAV. To maximize video quality and video transmission rate, the system's utility is defined as a linear summation of energy efficiency and QoE,

$$\eta[n] = \frac{\sum_{k=1}^{K} r_{O_k 2R}[n] + r_{R2B}[n]}{E_O + E_{comp} + E_R + E_F} + \sum_{k=1}^{K} QoE_k, \tag{17}$$

where $r_{O_k 2R}[n]$ is the offloading rate between the k^{th} OUAV and the RUAV, $r_{R2B}[n]$ is the video transmission rate between the RUAV and the GBS, E_O, E_{comp}, E_R, and E_F are the energy consumption of each link. QoE_k is the user QoE of the video after transcoding, which can be modeled as a natural logarithm function related to the bit rate as follows [32]

$$QoE_k = \ln\left(\frac{c_k}{c_{\min}}\right), \tag{18}$$

where $c_{\min}$ is the minimum bit rate threshold, $c_k \in c = \{c_1, c_2, \ldots, c_M\}$ is the video code rate.

As a result, The optimization problem is formulated as

$$\max_{\substack{p_k, p_R, \\ a_k, \boldsymbol{q}_R, D_k'}} \sum_{n=1}^{N} \eta[n], \tag{19a}$$

$$s.t. \frac{\|\boldsymbol{q}_R[n] - \boldsymbol{q}_R[n-1]\|}{\delta} \leq v_{\max}, \tag{19b}$$

$$p_k^{\min} \leq p_k[n] \leq p_k^{\max}, \tag{19c}$$

$$p_R^{\min} \leq p_R[n] \leq p_R^{\max}, \tag{19d}$$

$$\sum_{k=1}^{K} a_k[n] = 1, n \in \mathcal{N}, \tag{19e}$$

$$t_D \leq \delta, \tag{19f}$$

where equation (19a) denotes the optimization objective of the system, i.e., the system's utility. Equation (19b) is the mobility constraint of the RUAV, which means that the average speed cannot exceed the maximum speed of the RUAV. Equations (19c) and (19d) impose a limitation on the video transmit power for the OUAV and RUAV, respectively. Equation (19e) indicates the computational resources scheduling constraint, and the sum of computational resources allocated for each OUAV cannot exceed the total computational resources of the RUAV's CPU. Constraint (19f) ensures that the system latency does not exceed the time slot, as shown in Fig. 2.

IV. PROPOSED DDPG-BASED FRAMEWORK

The optimization problem mentioned above is a dynamic optimization problem, which demands a sequence of sequential decisions to optimize the objective while satisfying the constraints. Conventional resource allocation methods, such as static optimization and game theory, have difficulty solving the problem, because they attempt to figure out the near-optimal policy by maximizing the immediate reward in the current state. In the following sections, the problem (19) is modeled as an MDP, and then, we adopt DDPG to solve the MDP. For clarity, the frequently used notations related to the MAP and DDPG are summarized in Table II.

TABLE II

SUMMARY OF KEY NOTATIONS RELATED TO MDP AND DDPG.

Notations	**Description**	**Notations**	**Description**
$\mathcal{S}$	the state space	$\mathcal{A}$	the action space
r	the reward	$\mathcal{P}$	the transition distribution
γ	the discount factor	α_C	the learning rate of the critic
τ	the factor of soft update	B	the capacity of the replay buffer
α_A	the learning rate of the actor	Tanh	the activation function

A. Markov Decision Process

The MDP consists of $\langle \mathcal{S}, \mathcal{A}, r \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and r denotes the reward. The automatically learned UAV in RL is called the agent. The other elements of the MDP are described in detail as follows.

1) State space: Based on (3), (4), (7), (8) and (17), the distances between each OUAV and the RUAV and the distance between the RUAV and GBS are the most influential factor in the utility of the system. The distance between the communicating entities causes changes in the channel state information, which directly or indirectly affects the transmission rate, energy and QoE. The agent will choose the appropriate action policy according to the change of distance. Therefore, in the MDP, the distance is directly adopted as state. The state space $\mathcal{S}$ can be expressed as

$$\mathcal{S} = \{d_{O_1 2R}[n], d_{O_1 2R}[n], \ldots, d_{O_K 2R}[n], d_{R2B}[n]\}, \tag{20}$$

where $d_{O_1 2R}[n], d_{O_1 2R}[n], \ldots, d_{O_K 2R}[n]$ are the distances between each OUAV and the RUAV, respectively. $d_{R2B}[n]$ is the distance between the RUAV and GBS.

2) Action space: The action space $\mathcal{A}$ consists of five parts which is given by

$$\mathcal{A} = \{p_k[n], p_R[n], a_k[n], q_R[n], c\}, \tag{21}$$

where $p_k[n]$ is the transmit power of the OUAVs to the RUAV, $p_R[n]$ is the video transmit power of the RUAV to the GBS, $a_k[n]$ is the proportion of computational resources allocated to each OUAV, $q_R[n]$ is the moving direction of the RUAV, and c is the transcoding policy. Meanwhile, the constraints of (19b)-(19f) need to be satisfied.

3) Reward: The goal of the optimization problem is to maximize the system's utility in UAV-aided MEC network, so we define the system's utility within each time slot as the reward.

$$r_{s_n}^{a_n} = \rho_1 \frac{\sum_{k=1}^{K} r_{O_k 2R}[n] + r_{R2B}[n]}{E_O + E_{comp} + E_R + E_F} + \rho_2 \sum_{k=1}^{K} QoE_k, \tag{22}$$

where ρ_1 and ρ_2 are the prices of each system's utility item orderly obtained from network training [33]. $r_{s_n}^{a_n}$ implies that the reward at time slot n of the agent is generated based on state s_n and action a_n.

Since the agent can only obtain the system's utility of the current state, for the agent to make a policy that maximizes the cumulative reward, we define the state-value function to represent the cumulative utility from the current state to the termination state.

$$V(s_n) = \mathrm{E}\left\{ \sum_{i=n}^{N} \gamma^{N-i} r_{s_i}^{a_i} \right\}, \gamma \in [0,1], \tag{23}$$

where $\gamma \in [0,1]$ denotes the discount factor reflecting the importance of future reward values. If $\gamma = 0$, it means that only current rewards are considered; while the rewards in all states throughout the process have the same importance when $\gamma = 1$. (23) can be further rewritten as the *Bellman* equation,

$$V(s_n) = \mathrm{E}\left\{ r_{s_n}^{a_n} + \gamma V(s_{n+1}) \right\}, \gamma \in [0,1]. \tag{24}$$

Similarly, the *Bellman* equation for the action-value function is given by

$$Q^{\pi}(s_n, a_n) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left[r_{s_n}^{a_n} + \gamma \max_{a \in \mathcal{A}} Q^{\pi}(s_{n+1}, a_{n+1}) \right], \gamma \in [0,1], \tag{25}$$

where $\pi(a \mid s)$ is the mapping of states to actions, representing the action selection policy of the agent in different environments.

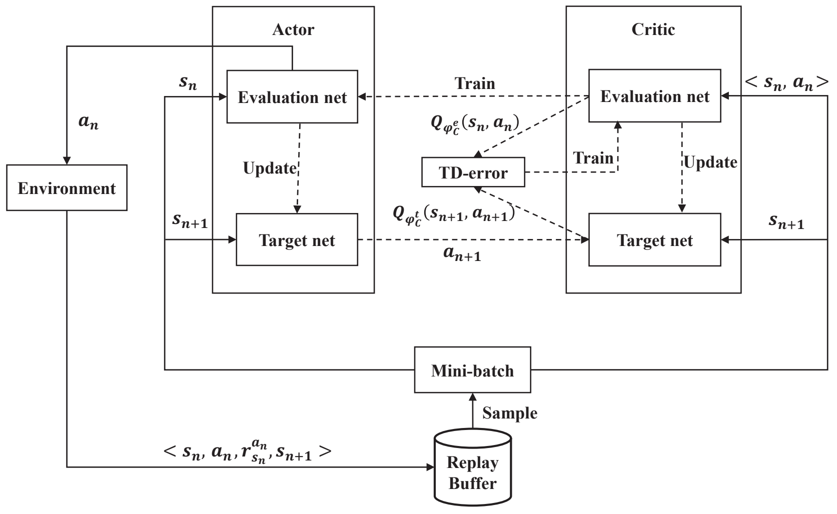


Fig. 3. The structure of DDPG-based framework.

B. DDPG-based Framework

Considering that the state-action space of is continuous, the DDPG algorithm in DRL is adopted to achieve continuous control of the UAV. Since DDPG is a policy gradient algorithm, which derives the value expectation without optimal action selection, it is applicable to the continuous state-action space. Meanwhile, DDPG is an off-policy, which is beneficial for UAVs to train through off-line data. In addition, DDPG is also a deterministic policy, i.e., the same policy takes the same action in the same state. Therefore, the DDPG algorithm has higher learning efficiency, better convergence, and stability.

The structure of the DDPG-based framework is shown in Fig. 3. The DDPG algorithm consists of two deep neural networks (DNNs), an actor network and a critic network, to solve the problem of Q-value instability during learning. The actor network outputs continuous actions, and then the continuous actions are fed into the critic network along with the states. The actor network approximates the action, thus eliminating the need to find the action that maximizes the Q-value function in the case of non-convex optimization. As shown in Fig. 3, there are four network models in the DDPG: two structurally identical actor networks, the actor evaluation network and the actor target network, and two structurally identical critic networks, the critic evaluation network and the critic target network. The critic and actor networks are elaborated in the following.

1) Critic: Similar to the structure of DQN, critic contains the target network and the evaluation network. The evaluation network computes the evaluation of the action-value function based on the states and actions.

$$Q_{\varphi_C^e}\left(s_n, a_n\right) = \omega_C^s s_n + \omega_C^a a_n + b_C^e \approx Q\left(s_n, a_n\right), \tag{26}$$

where φ_C^e is the evaluation network parameter of critic, ω_C^s and ω_C^a are the neural network weights, and b_C^e is the neural network bias. The target network calculates the target value of the action-value function based on $r_{s_n}^{a_n}$ and s_{n+1}, and action a_{n+1} is generated from the target network of the actor.

$$Q_{\varphi_C^t}\left(s_n, a_n\right) = \mathbb{E}\left[r_{s_n}^{a_n} + \max_{a_{n+1}\in\mathcal{A}} \gamma Q_{\varphi_C^t}\left(s_{n+1}, a_{n+1}\right)\right], \tag{27}$$

where φ_C^t is the target network parameter of critic. The loss function of the critic evaluation network is defined as the mean squared error between the target value and the evaluation value, which can be written as

$$\operatorname{loss}\left(\varphi_C^e\right) = \left(r_{s_n}^{a_n} + \gamma Q_{\varphi_C^t}\left(s_{n+1}, a_{n+1}\right) - Q_{\varphi_C^e}\left(s_n, a_n\right)\right)^2. \tag{28}$$

The parameters of evaluation network are updated by gradient descent which is given by

$$\varphi_C^e = \varphi_C^e + \alpha_C \nabla_{\varphi_C^e} \operatorname{loss}\left(\varphi_C^e\right), \tag{29}$$

where α_C is the learning rate of the critic. Instead of updating the parameters by gradient descent, the parameters of target network are replaced by the parameters of evaluation network by

$$\varphi_C^t = (1-\tau)\varphi_C^t + \tau\varphi_C^e, \tag{30}$$

where τ is a positive number less than 1. Since the target network is not updated in a single step, the magnitude of the target network can be controlled by (30) to ensure the stability of the training.

2) Actor: Similar to the critic, the actor contains the target network and the evaluation network. The target network calculates the target action and the target value of the action-value function. The evaluation network is utilized to calculate the performed action to obtain the maximum feedback from the critic. The larger the $Q_{\varphi_C^e}$ obtained from the feedback, the smaller the loss. Therefore, we take the opposite of the mean of the action-value returned by the evaluation network as the loss function which is given by

$$\operatorname{loss}\left(\varphi_A^e\right) = -\frac{1}{B}\sum_{i=1}^{B} Q_{\varphi_C^e}\left(s_i, a_i\right), \tag{31}$$

where B is the capacity of the replay buffer. Similar to the critic network, the parameters of the evaluation and target networks are updated as follows

$$\varphi_A^e = \varphi_A^e + \alpha_A \nabla_{\varphi_A^e} \operatorname{loss}\left(\varphi_A^e\right), \tag{32}$$

$$\varphi_A^t = (1-\tau)\varphi_A^t + \tau\varphi_C^e, \tag{33}$$

where α_A is the learning rate of the actor.

In the end, to enable the actor to select the continuous action, we adopt the $Tanh$ function as an activation function for the output layer of the actor network. We limit the output value range between 0 and 1 and multiply it with the maximum value in the range required by the environment as the final output action,

$$\left[a_1, a_2, \ldots, a_Y\right] = \operatorname{Tanh}\left(\omega_A^e s_n + b_A^e\right) \cdot \mathrm{G}_a, \tag{34}$$

where Y is the dimension of the action space. ω_A^e and b_A^e are the neural network parameters, i.e., $\varphi_A^e = [\omega_A^e, b_A^e]$, ω_A^e is the neural network weight, and b_A^e is the neural network bias. G_a is the maximum value of the output action, which is utilized to control the output of the neural network to meet the environmental requirements. Since DDPG is a deterministic policy, unlike the conventional random policy, which is inherently exploratory, an additional exploration policy needs to be set up. In this paper, we introduce the behavior policy [34], which adds random noise ϑ_n to the output of the action from the actor evaluation network to change the deterministic-valued actions performed by the agent into random-valued actions $a'_n = a_n + \vartheta_n$.

The samples for RL are collected in chronological order and are strongly correlated. Therefore, the samples do not satisfy the requirement of an independently identical distribution for neural network training. To tackle this challenge, an experience replay buffer is needed to break the correlation between the samples. The replay buffer is a finite-size buffer that stores sample tuples $\left(s_n, a_n, r_{s_n}^{a_n}, s_{n+1}\right)$ obtained from the interaction between the agent and the environment. When the parameters of the actor-critic network need to be updated, a mini-batch of samples is randomly selected from the replay buffer by uniform random sampling to accelerate the convergence of the neural network.

C. Overall Algorithm

The specific approach for the issue is summarized in Algorithm 1. At the beginning of the DDPG algorithm, the replay buffer $\mathcal{D}$, the weights of actor and critic in the RUAV are initialized. Notice that the training procedure comprises of E episodes, each of which consists of N steps. Generally, at the beginning of each episode, we first initialize the state s_n and action a_n. Then, in each step n, the action of RUAV at state s_n is generated by its online actor network with a random noise ϑ_n. Based on the action taken above, the RUAV sets its moving direction and transmission power. Additionally, resource allocation and transcoding policy are dynamically selected. Then, the tuple $< s_n, a'_n, r_{s_n}^{a_n}, s_{n+1} >$ is stored in the replay buffer $\mathcal{D}$. After sampling from the replay buffer $\mathcal{D}$, the online networks of actor and critic can be updated. The target networks of actor and critic are updated in (30) and (33).

In general, the computational complexity of the DDPG algorithm is related to the dimensions of the input state and action, the neuron numbers in each neural network layer, and the layer numbers

Algorithm 1 DDPG utility-oriented optimization algorithm

```
begin
    Initialize replay buffer $\mathcal{D}$, the number of steps $N$, and training episodes $E$; structures of networks, i.e., $\varphi_C^e$, $\varphi_C^t$, $\varphi_A^e$, $\varphi_A^t$, $\alpha_C$, $\alpha_A$, $\gamma$ and $\tau$.
    for e=1:E do
        Initialize state space $\mathcal{S}$, action space $\mathcal{A}$, a random noise $\vartheta_n$ for action exploration;
        for n=1:N do
            Input $s_n$ into the actor network and generate the action $a_n$;
            Select action $a_n' = a_n + \vartheta_n$ according to the current policy and exploration noise;
            Execute action $a_n'$, observe new state $s_{n+1}$ and reward $r_{s_n}^{a_n}$;
            Update replay buffer: $\mathcal{D} \leftarrow < s_n, a_n', r_{s_n}^{a_n}, s_{n+1} > \cup \mathcal{D}$;
            if replay buffer is full then
                Sample a mini-batch from $\mathcal{D}$;
                Update actor and critic network parameters:
                $\varphi_C^e = \varphi_C^e + \alpha_C \nabla_{\varphi_C^e} \operatorname{loss}(\varphi_C^e)$, $\varphi_A^e = \varphi_A^e + \alpha_A \nabla_{\varphi_A^e} \operatorname{loss}(\varphi_A^e)$
                for j=1:t do
                    Update target actor and critic network parameters:
                    $\varphi_C^t = (1-\tau)\varphi_C^t + \tau\varphi_C^e$, $\varphi_A^t = (1-\tau)\varphi_A^t + \tau\varphi_C^e$
                end
            end
        end
    end
end
```

of the neural network [6]. Meanwhile, both actor and critic networks should be considered. When the neural network parameters converge after N epochs and E episodes, the computational complexity can be expressed as $\mathcal{O}\left(NE\left(\sum_{g=0}^{G_A-1} u_A^g u_A^{g+1} + \sum_{g=0}^{G_C-1} u_C^g u_C^{g+1}\right)\right)$, where G_A and G_C denote the layer numbers of the actor and critic networks and u_A^g and u_C^g denote the neuron numbers in the actor and critic networks in the g^{th} layer, respectively.

V. SIMULATION RESULTS AND DISCUSSIONS

In this section, numerical results are provided to evaluate the performance of the proposed DDPG algorithm. First, we describe the simulation parameters. Then, we present benchmark schemes, followed by results and discussions.

A. Simulation Setup

This paper investigates the effectiveness of a UAV-aided video transmission system based on MEC to improve utility performance. We, therefore, choose a basic system model to study the novel scheme, where a RUAV and two OUAVs (i.e., K = 2) are deployed with a maximum flight speed $v_{max} \in \{10, 20, 30\}$ m/s. However, the proposed scheme and algorithm can be applied to a more complex case with multiple OUAVs. Each OUAV follows a fixed circular trajectory, with a flight radius r_O = 200 m. In the actual scenario we consider, one GBS is located in [0, 0, 0] m. It is assumed that the flight height of RUAV $H_R \in \{80, 90, 100, 110, 120\}$ m. We define the flight height ratio (FHR) of the RUAV and OUAVs as FHR = $\frac{H_R}{H_k} \in \{0.5, 1, 2, 3\}$.

For default system parameters, the bandwidth of the system is $\omega \in \{700, 800, 900\}$ KHz [26]. The channel gain is $\beta_0 = 1.42 \times 10^{-4}$ at the reference distance d_0 = 1 m and the noise power spectral density is set as $N_0 = 1.99 \times 10^{-10}$ mW. Considering a complex suburban environment, the S-curve parameters are set as a = 4.88 and b = 0.43 under f_c = 2 GHz with the excessive path-loss coefficient η_{LoS} = 0.1 dB and η_{NLoS} = 21 dB [28]. The transmit power of the UAV is $P \in$ [0, 200] mW. The minimum time slot of the system is δ = 1 s. Meanwhile, we select a video with an original code rate of 2.75 Mbps (1080p) as the original video [30]. The RUAV transcodes the original video to five levels, i.e., $c \in \{0.3, 0.8, 1.3, 1.8, 2.3\}$ Mbps. The correlation coefficients of the CPU calculation cycle are ξ = 1.54 and ς = 0.08 [30], and the energy consumption coefficients of the CPU are j_R = 10^{-27} and s_R = 3 [31]. The maximum CPU frequency of the RUAV is f_{max} = 10 GHz. And other simulation parameters of propulsion energy consumption are referred to [12].

B. Benchmark Schemes

To verify the effectiveness of the proposed algorithm, i.e., the DDPG algorithm, a widely used DQN algorithm [17] is also utilized to search the near-optimal policy for MDP. However, DQN can only control discrete actions resulting in the loss of many actions in the discretization of continuous variables. Furthermore, the AC algorithm [18] is also simulated, which is similar to DDPG in structure, but the AC updates the neural network parameters continuously. The actor generates the mean and variance when calculating the policy and samples the normal distribution to realize the decision of continuous action. For reinforcement learning settings, we set E = 400 episodes with N = 400 steps each in the training stage. Due to the continuous policy and the environment with no end point, we manually abort the training process and reset the OUAVs, RUAV, and environment every episode. As the replay buffer has a capacity of 20,000 and training will not start until the buffer is full, the random policy is utilized for the first 20,000 steps and training is started at each step afterwards. The actor and critic neural networks of RUAV are implemented based on the Tensorflow framework, with $\alpha_A \in \{1 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}\}$, $\alpha_C \in \{2 \times 10^{-3}, 2 \times 10^{-4}, 2 \times 10^{-5}\}$, and $\gamma \in \{0.8, 0.9, 1\}$.

C. Results and Discussions

The effectiveness and convergence of the DDPG algorithm are demonstrated and discussed in the following. The learning curves of different learning rates α_A and α_C and discount factor γ

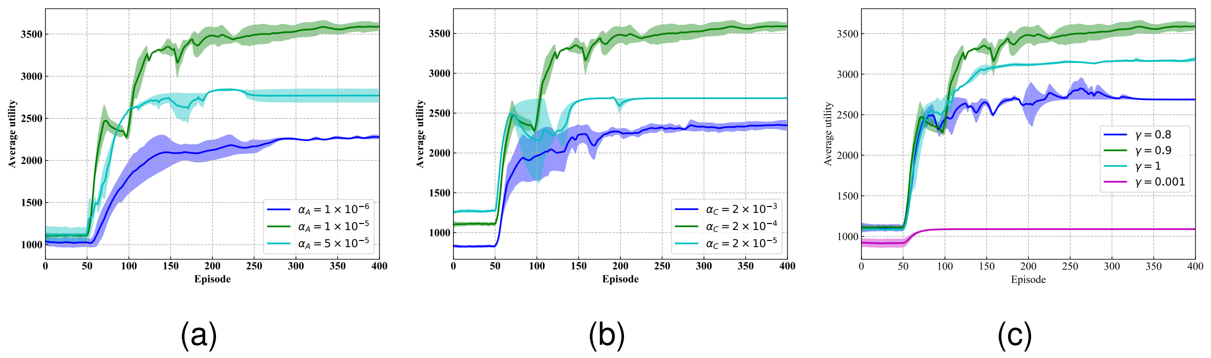


Fig. 4. The convergence performance. (a) With different actor learning rates. (b) With different critic learning rates. (c) With different discount factors.

are shown in Fig. 4. The changing trend of the three algorithms, as well as the corresponding results during the training period, are shown in Fig. 5, Fig. 6, and Fig. 7. The RUAV's flight trajectories are shown in Fig. 8.

1) Impact of DRL Parameters: As shown in Fig. 4(a), the learning rate of the actor, i.e., α_A, has an impact on the effectiveness and convergence. The dark lines indicate the mean value of the average utility, while the lighter areas take into account the standard error based on the mean value and reflect the differences in the curves [35]. DDPG converges at approximately 350 episodes (400 steps per episode) and achieves the highest utility in all learning rates when the learning rate is 1×10^{-5}. If the learning rate increases to 5×10^{-5}, the learning process fluctuates considerably and achieves lower utility after convergence. In contrast, the fluctuation of the training process is reduced when the learning rate is equal to 1×10^{-6}, but the utility decreases significantly. Therefore, in all subsequent simulations, α_A is fixed at 1×10^{-5}.

Fig. 4(b) shows the comparison of convergence with different learning rates of the critic, i.e., α_C. If the learning rate is 2×10^{-4}, DDPG reaches stability at approximately 350 episodes. If the learning rate decreases to 2×10^{-5}, DDPG achieves lower utility after convergence. For $\alpha_C = 2 \times 10^{-3}$, the learning speed will become slower, not reaching stability until approximately 300 episodes, and the average utility obtained is relatively low. Therefore, α_C is fixed at 2×10^{-4} in all subsequent simulations for the highest utility.

Fig. 4(c) shows the comparison of convergence performance with different discount factors γ. The discount factor indicates the importance of future returns, a discount factor of 1 indicates that all future returns have the same importance, and 0 indicates that only the returns in the current state are considered. As shown in Fig. 4(c), when γ = 0.001, i.e., approaching 0, the reward does not rise well and remains essentially the same as the initial reward value, which is caused by its failure to focus on the future return. If the discount factor is 0.9, DDPG reaches stability at approximately 350 episodes and maintains stable utility in the subsequent training process. If the discount factor is increased to 1, DDPG achieves lower utility after convergence. This is because that the neural network focuses too much on the reward of future states, which affects the correct judgment of the agent in the current state. If the discount factor is decreased to 0.8, the learning suffers a violent oscillation and reaches stability for more than 310 episodes. This means that the lower discount factor reduces the neural network's perception of future returns, which affects the agent's ability to make policies that maximize utility.

2) Evolution of Different Algorithms: Fig. 5 shows the average utility of the different algorithms when H_R = H_k = 80 m. Since the neural network structures of DDPG and AC are similar, the same learning rate is used in the training process. It is observed that DQN has the lowest average utility after convergence, AC has a slight improvement, and DDPG achieves the highest utility. This is because that DQN can only control discrete actions, and the optimization variables of the system include a variety of continuous actions, such as the power allocation of the UAVs and the RUAV's flight trajectory. A large number of efficient action choices are lost in the process of discretizing the above continuous variables. AC has a significantly lower rate of increase in utility and achieves a utility slightly lower than DDPG. The apparent reason is the instability of the training process caused by the correlation between each sample. In addition, the loss function of critic is involved in the update of the actor neural network, which reduces

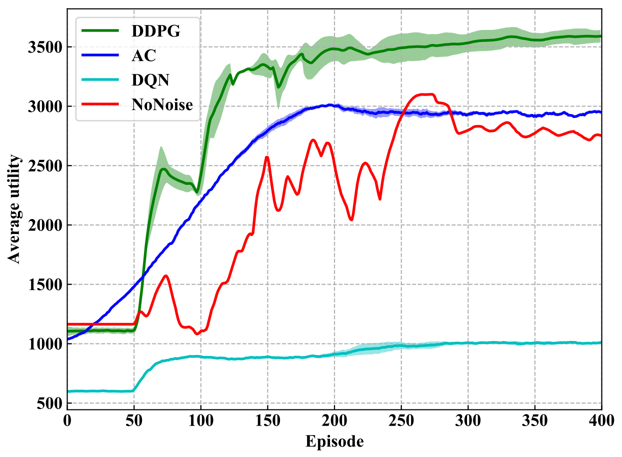


Fig. 5. The utility performance of different algorithms.

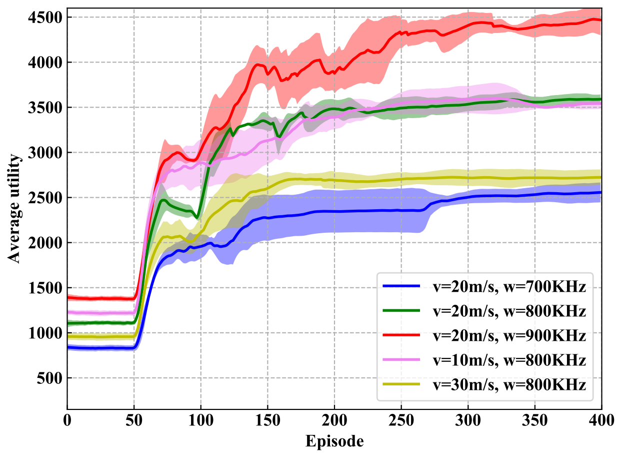
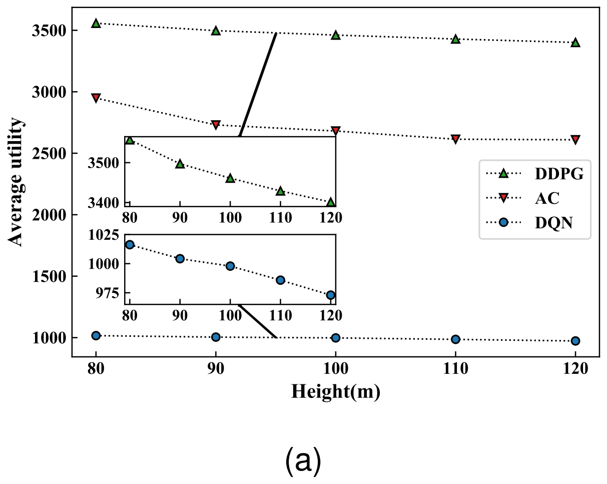


Fig. 6. The utility performance of different bandwidths and maximum flight speeds.

the convergence speed of the algorithm. To solve this problem, the training samples are sampled by certain priorities in the replay buffer, which improves the stability of training, and the actor directly optimizes the value of states, making the algorithm easy to converge. The algorithm without random noise (*NoNoise* in Fig. 5) performs worse than the proposed DDPG algorithm, demonstrating the efficiency of the behavior policy.

3) Impact of Maximum Flight Speed and Bandwidth: We obtain the average utility for



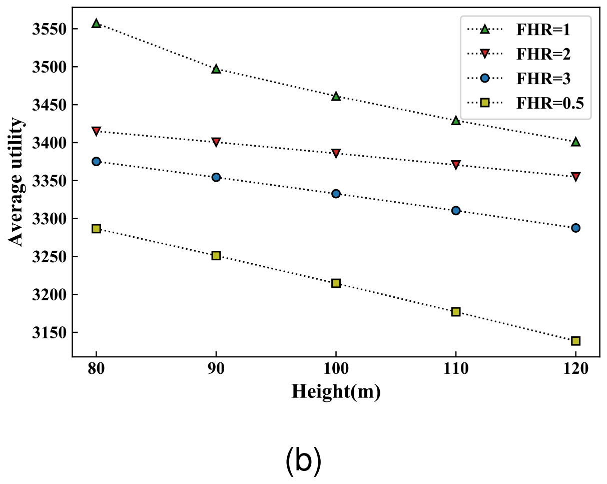


Fig. 7. The utility performance w.r.t the fight height. (a) With different algorithms. (b) With different FHR.

different bandwidths and maximum flight speeds in Fig. 6. Interestingly, the average utility obtained is surprisingly consistent for different maximum speeds of 10 m/s and 20 m/s. However, the average utility performs poorly when the maximum flight speed reaches 30 m/s, which is an interesting issue that has been studied in depth in [12]. Zeng *et al.* verified that as the flight speed increased, the propulsion power consumption firstly decreased and then increased with the flight speed. Two particular UAV speeds with high practical interests, i.e., the maximum-endurance speed (v_{me}, nearly 10 m/s) and the maximum-range speed (v_{mr}, nearly 20 m/s) were found numerically. With v_{ME}, and v_{mr}, the propulsion power consumption is close to the optimal value compared to 30 m/s. Considering the longer flight distance and shorter task completion time, let v_{max} = 20 m/s. Additionally, the effect of bandwidth on utility is enormous, and increasing bandwidth brings an increase in utility. Nevertheless, too high bandwidth places higher demands on the hardware and risks wasting resources. ω = 800 KHz is regarded as a trade-off scheme.

4) Impact of Height of UAVs: Fig. 7(a) shows the average utility versus flight height for different algorithms when H_R = H_k. From the figure, we have the following conclusions. First, at the same flight height, the average utility of DDPG is the highest, followed by AC and DQN.

The reasons are described in Fig. 5. Second, the average utility gradually decreases as the flight height of the RUAV increases. As the flight height of the UAV increases, the channel conditions between the UAV and GBS deteriorate, thus affecting the communication quality and utility. Therefore, more transmit power needs to be consumed to ensure that the channel capacity can complete the transmission of the video.

Fig. 7(b) shows that the flight height ratio (FHR) of the RUAV and OUAVs has an impact on the average utility in our DPPG algorithm. When the RUAV and OUAVs are at the same height, the average utility is the highest. Moreover, it can be seen that the average utility increases rapidly with the height of the OUAVs and the RUAV approaches. The height difference of the OUAVs and the RUAV directly affects the transmit power between them according to (3) and (7). Similar to Fig. 7(a), the average utility gradually decreases as the flight height of the RUAV increases. In actual scenario, the flight height of OUAVs may be reduced considering the clarity of the video, so it makes sense to consider the change in utility caused by the height difference between OUAVs and RUAV.

5) UAV Trajectory Analysis: Fig. 8(a) demonstrates the trajectories of RUAV obtained by DDPG, DQN, and AC algorithms with the UAV flight duration T = 126 s and T = 63 s. The flight height is fixed at H_R = H_k = 80 m. The initial position of the RUAV is set as (-800, 0). Two OUAVs follow a clockwise circular trajectory and the centers of the trajectory are (-1200, 0) and (-1200, -400). Because of the long distance from the GBS, the RUAV with DDPG tries to move toward the GBS within the communication range, which is not achieved with DQN and AC algorithms. From Fig. 8, it can be seen that the RUAV flies similar trajectories under different flight duration. The result shows that the RUAV keeps the same ”thought process” in seeking a trade-off between receiving video from OUAVs and forwarding video to the GBS. Supplementarily, when T = 126 s, the RUAV possesses a longer learning period to adapt to changes in the channel environment several times.

The flight trajectory with DDPG can be observed in detail in Fig. 8(b) with some key points

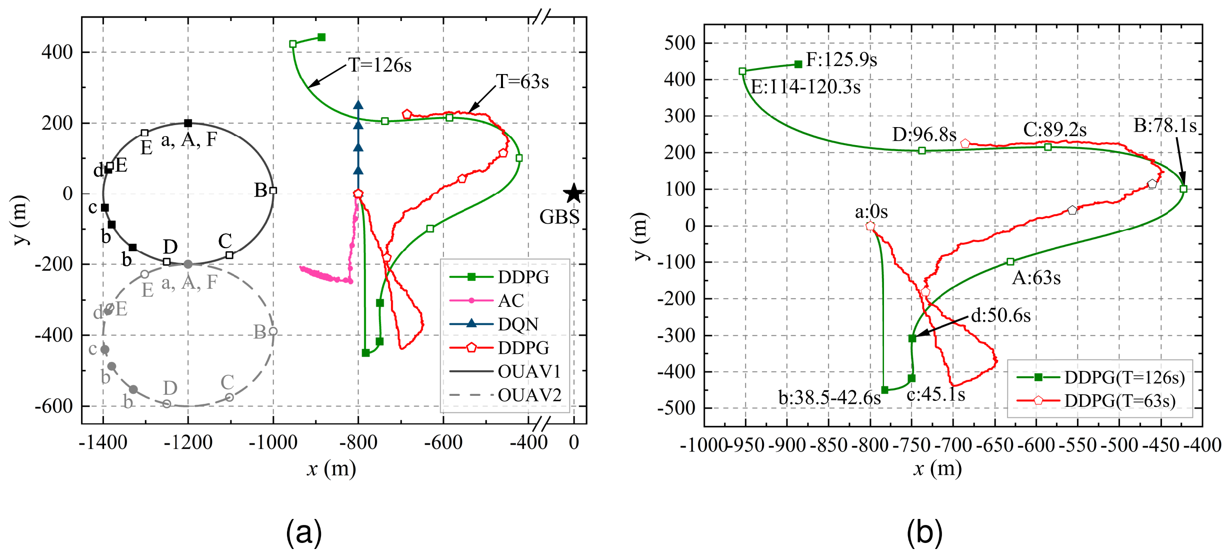


Fig. 8. The UAV flight trajectories. (a) With different algorithms. (b) With DDPG in detail.

that has been marked. During the OUAVs' two clockwise flights ($T = 126$ s), the RUAV's trajectory with DDPG can be divided into six segments:

- From $\boldsymbol{a}$ to $\boldsymbol{b}$, the RUAV follows the OUAVs downward to receive the original video better. When the OUAVs change the state to fly upward, the RUAV briefly hovers at $\boldsymbol{b}$ to obtain the intent of the OUAVs.
- From $\boldsymbol{b}$ to $\boldsymbol{d}$, the RUAV flies slowly upward and adjusts the distance between itself and the GBS to communicate better. Then, RUAV flies rapidly upward to follow the OUAVs flying upward to $\boldsymbol{d}$.
- From $\boldsymbol{d}$ to $\boldsymbol{A}$, when the OUAVs start moving to the right, the RUAV follows upward and approaches the GBS significantly to the right through autonomous learning. Then the OUAVs commence a new round of flights.
- From $\boldsymbol{A}$ to $\boldsymbol{B}$, the RUAV continuously approaches the GBS and approaches the downlinked OUAVs by flying upward.
- From $\boldsymbol{B}$ to $\boldsymbol{D}$, the RUAV realizes that it is too far from the OUAVs and compensates by

flying to the left, even slightly downward.

- From $\boldsymbol{D}$ to $\boldsymbol{E}$, the RUAV follows the OUAVs upward. After hovering for a moment, the RUAV flies to $\boldsymbol{F}$ like $\boldsymbol{d}$ to $\boldsymbol{A}$ and completes the whole mission at T = 126 s.

VI. Conclusion

This paper considered a utility-oriented optimization scheme for a UAV-aided video transmission system based on MEC. In the constructed system, the OUAVs shot videos in a fixed trajectory, and the RUAV transcoded the videos and sent the transcoded videos to GBS. We maximized the utility of the communication system by jointly optimizing the power allocation, the transcoding policy, the computational resources allocation, and the flight trajectory of the RUAV. Considering the dynamic characteristics of wireless channels, the above-mentioned problem was modeled as an MDP and solved by the DDPG algorithm. The simulation results demonstrated the effectiveness and superiority of the DDPG algorithm compared with other approaches. The utility of the system and the flight trajectory of the UAV have also been demonstrated. In future research, we will consider the issues of multi-agent and physical layer security.

References

[1] L. A. b. Burhanuddin, X. Liu, Y. Deng, U. Challita, and A. Zahemszky, “Qoe optimization for live video streaming in uav-to-uav communications via deep reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5358–5370, 2022.

[2] Q. Wu, Y. Zeng, and R. Zhang, “Joint trajectory and communication design for multi-uav enabled wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[3] M. Pinkney, D. Hampel, and S. DiPierro, “Unmanned aerial vehicle (uav) communications relay,” in *Proceedings of MILCOM ’96 IEEE Military Communications Conference*, vol. 1, 1996, pp. 47–51.

[4] Y. Zeng, R. Zhang, and T. J. Lim, “Throughput maximization for uav-enabled mobile relaying systems,” *IEEE Transactions on Communications*, vol. 64, no. 12, pp. 4983–4996, 2016.

[5] C. Liu, H. Zhang, H. Ji, and X. Li, “Mec-assisted flexible transcoding strategy for adaptive bitrate video streaming in small cell networks,” *China Communications*, vol. 18, no. 2, pp. 200–214, 2021.

[6] Y. Liu, J. Yan, and X. Zhao, “Deep reinforcement learning based latency minimization for mobile edge computing with virtualization in maritime uav communication network,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 4225–4236, 2022.

[7] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, “Multi-agent deep reinforcement learning for task offloading in uav-assisted mobile edge computing,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[8] Y. Xu, T. Zhang, D. Yang, and L. Xiao, “Uav-assisted relaying and mec networks: Resource allocation and 3d deployment,” in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.

[9] B. Liu, Y. Wan, F. Zhou, Q. Wu, and R. Q. Hu, “Resource allocation and trajectory design for miso uav-assisted mec networks,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4933–4948, 2022.

[10] N. Zhao, Z. Liu, and Y. Cheng, “Multi-agent deep reinforcement learning for trajectory design and power allocation in multi-uav networks,” *IEEE Access*, vol. 8, pp. 139 670–139 679, 2020.

[11] N. Zhao, Y.-C. Liang, D. Niyato, Y. Pei, M. Wu, and Y. Jiang, “Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

[12] Y. Zeng, J. Xu, and R. Zhang, “Energy minimization for wireless communication with rotary-wing uav,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[13] Y. Wang, M. Chen, C. Pan, K. Wang, and Y. Pan, “Joint optimization of uav trajectory and sensor uploading powers for uav-assisted data collection in wireless sensor networks,” *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 214–11 226, 2022.

[14] T. Zhang, G. Liu, H. Zhang, W. Kang, G. K. Karagiannidis, and A. Nallanathan, “Energy-efficient resource allocation and trajectory design for uav relaying systems,” *IEEE Transactions on Communications*, vol. 68, no. 10, pp. 6483–6498, 2020.

[15] L. Xiao, Y. Xu, D. Yang, and Y. Zeng, “Secrecy energy efficiency maximization for uav-enabled mobile relaying,” *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 1, pp. 180–193, 2019.

[16] Y. Zeng, Q. Wu, and R. Zhang, “Accessing from the sky: A tutorial on uav communications for 5g and beyond,” *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, 2019.

[17] C. Zhan and Y. Zeng, “Energy minimization for cellular-connected uav: From optimization to deep reinforcement learning,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 7, pp. 5541–5555, 2022.

[18] M. Nasr-Azadani, J. Abouei, and K. N. Plataniotis, “Single- and multiagent actor–critic for initial uav’s deployment and 3-d trajectory design,” *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 15 372–15 389, 2022.

[19] Q. Hou, Y. Cai, Q. Hu, M. Lee, and G. Yu, “Joint resource allocation and trajectory design for multi-uav systems with moving users: Pointer network and unfolding,” *IEEE Transactions on Wireless Communications*, vol. 22, no. 5, pp. 3310–3323, 2023.

[20] Q. Zhang, J. Miao, Z. Zhang, F. R. Yu, F. Fu, and T. Wu, “Energy-efficient video streaming in uav-enabled wireless networks: A safe-dqn approach,” in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–7.

[21] C. Zhan and R. Huang, “Energy efficient adaptive video streaming with rotary-wing uav,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 8040–8044, 2020.

[22] I. Medeiros, A. Boukerche, and E. Cerqueira, “Swarm-based and energy-aware unmanned aerial vehicle system for video delivery of mobile objects,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 766–779, 2022.

[23] Y. Liu, F. R. Yu, X. Li, H. Ji, H. Zhang, and V. C. M. Leung, “Joint access and resource management for delay-sensitive transcoding in ultra-dense networks with mobile edge computing,” in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[24] Z. Li, R. Xie, Q. Jia, and T. Huang, “Energy-efficient joint caching and transcoding for http adaptive streaming in 5g networks with mobile edge computing,” in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[25] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, “Placement optimization of uav-mounted mobile base stations,” *IEEE Communications Letters*, vol. 21, no. 3, pp. 604–607, 2017.

[26] G. Zhang, Q. Wu, M. Cui, and R. Zhang, “Securing uav communications via joint trajectory and power control,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1376–1389, 2019.

[27] A. A. Khuwaja, Y. Chen, N. Zhao, M.-S. Alouini, and P. Dobbins, “A survey of channel modeling for uav communications,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2804–2821, 2018.

[28] Y. Zhang, Z. Mou, F. Gao, J. Jiang, R. Ding, and Z. Han, “Uav-enabled secure communications by multi-agent deep reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 11 599–11 611, 2020.

[29] A. Al-Hourani, S. Kandeepan, and S. Lardner, “Optimal lap altitude for maximum coverage,” *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[30] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon, “Transcoding live adaptive video streams at a massive scale in the cloud,” in *Proceedings of the 6th ACM Multimedia Systems Conference*, ser. MMSys ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 49–60. [Online]. Available: https://doi.org/10.1145/2713168.2713177

[31] B. Dai, J. Niu, T. Ren, Z. Hu, and M. Atiquzzaman, “Towards energy-efficient scheduling of uav and base station hybrid enabled mobile edge computing,” *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 915–930, 2021.

[32] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, “Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction,” in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 272–285.

[33] Z. Zhang, Q. Zhang, J. Miao, F. R. Yu, F. Fu, J. Du, and T. Wu, “Energy-efficient secure video streaming in uav-enabled wireless networks: A safe-dqn approach,” *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 4, pp. 1892–1905, 2021.

[34] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction second edition,” *Cambridge, Massachusetts*, vol. 1, pp. 103–110, 2018.

[35] H. Shi, L. Shi, M. Xu, and K.-S. Hwang, “End-to-end navigation strategy with deep reinforcement learning for mobile robots,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2393–2402, 2019.