Computer Networks 280 (2026) 112164

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

DRL-based privacy-preserving video streaming task offloading under energy constraints in MEC

Hongyi Chen [a], Yun Peng [a], Chen Wei [a], Tao Meng [a], Wei Ai [a,*], Zhixiong He [b], Keqin Li [c]

[a] College of Computer and Mathematics, Central South University of Forestry and Technology, 410004, Changsha, China
[b] College of Economics and Management, Central South University of Forestry and Technology, 410004, Changsha, China
[c] Department of Computer Science, State University of New York, 12561, New Paltz, New York, USA

ARTICLE INFO

Keywords:
MEC
Task offloading
DRL
Video processing tasks

ABSTRACT

With the rapid proliferation of video surveillance, autonomous driving, and smart home applications, video task offloading in Mobile Edge Computing (MEC) has attracted growing research interest. Compared with traditional cloud computing, MEC reduces latency and energy consumption by deploying computational resources closer to end devices. However, it also introduces two fundamental challenges: privacy leakage from untrusted edge servers and resource constraints that cause dynamic fluctuations in latency, energy consumption, and Quality of Service (QoS). Existing optimization and learning-based methods often fail to ensure privacy within energy limits or to balance multiple conflicting objectives under dynamic MEC conditions. To overcome these challenges, this paper addresses the core difficulty of jointly modeling and optimizing multi-objective (accuracy, latency, energy) and multi-constraint (privacy, energy cap) problems in MEC-based video task offloading, where privacy protection and resource limitations coexist. We formulate this as a Constrained Markov Game (CMG) and propose a novel Constrained Policy Optimization Multi-Agent Deep Deterministic Policy Gradient (CPO-MADDPG) algorithm. Extensive simulations demonstrate that CPO-MADDPG effectively mitigates privacy leakage, maintains energy constraints, and minimizes latency while maximizing video analysis accuracy. The proposed framework provides a unified and scalable solution to the long-standing trade-off between privacy protection and resource efficiency in dynamic MEC environments.

1. Introduction

Recent advancements in video surveillance, autonomous driving, and smart homes have significantly improved video data acquisition, processing, and transmission technologies. However, these developments have also introduced challenges related to computing power, energy efficiency, and real-time responsiveness of end devices [1,2]. Traditional cloud computing models face limitations due to the need to transmit large video data volumes, resulting in high latency and bandwidth issues [3,4], which hinder real-time video processing. Edge computing addresses these challenges by placing computing resources closer to end users, enabling task offloading to edge servers [5–7]. This reduces the computational load on mobile devices, improving response times, processing performance, and energy efficiency, thus extending battery life. However, the growing use of task offloading in mobile edge computing (MEC) brings complex technical challenges in its deployment [8–10].

In MEC environments, offloading video data to potentially untrusted edge servers introduces significant privacy risks. The server could infer sensitive information, such as the device's location and usage patterns, from task characteristics like frame rate and resolution [11]. To address this, privacy-preserving schemes have been proposed. Wang et al. [12] introduced a differential privacy-based mechanism for protecting vehicle speed and location in the Internet of Vehicles. Xu et al. [13] proposed a privacy-aware offloading scheme using Privacy Entropy to safeguard data. However, these solutions face a trade-off between privacy and performance–excessive noise can degrade accuracy, while weak constraints risk exposing user data. Thus, achieving adaptive privacy protection without compromising real-time performance remains an open challenge in MEC-based video task offloading.

* Corresponding author.
E-mail addresses: chenhongyi@csuft.edu.cn (H. Chen), pengyun@csuft.edu.cn (Y. Peng), weichen@csuft.edu.cn (C. Wei), mengtao@hnu.edu.cn (T. Meng), aiwei@hnu.edu.cn (W. Ai), hezhixiong@hnu.edu.cn (Z. He), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.comnet.2026.112164
Received 2 December 2025; Received in revised form 22 January 2026; Accepted 24 February 2026
Available online 26 February 2026

Another major challenge in MEC-based video offloading is the resource-constrained nature of the environment. Multiple heterogeneous devices compete for limited computing, communication, and spectrum resources [14], causing fluctuations in latency, transmission reliability, and energy consumption, which degrade Quality of Experience. Researchers have proposed several solutions to address these issues. Zhao et al. [15] introduced a federated deep reinforcement learning approach that optimizes resources across multiple edge cells, enabling secure video offloading. In [16], Zhao et al. combined physical layer security with deep reinforcement learning to ensure communication security and improve Quality of Service during offloading. However, these methods typically focus on a single objective (e.g., delay or energy) and lack an integrated approach to address multi-objective, multi-constraint optimization, where latency, energy, and privacy must be balanced under resource constraints.

To tackle the above-mentioned challenges of privacy leakage and resource constraints, we extend the traditional multi-agent markov game framework into a constrained markov game (CMG) model to characterize the multi-objective and multi-constraint nature of MEC-based video task offloading. Within this framework, we propose a Constrained Policy Optimization Multi-Agent Deep Deterministic Policy Gradient (CPO-MADDPG) algorithm that jointly optimizes latency, energy, and privacy objectives. Specifically, we design a dual-channel policy gradient mechanism that separates the learning of task rewards and constraint costs into two interactive channels. These channels are dynamically balanced through Lagrange multipliers, achieving projection-free constrained learning and adaptive control of energy consumption.Building upon this mechanism, CPO-MADDPG maintains the decentralized actor structure of MADDPG but replaces its vanilla critic with a two-headed critic that simultaneously estimates the accuracy reward and energy cost. The hard energy cap is reformulated as a soft constraint, whose penalty weight is automatically adjusted via the same Lagrange multiplier. Furthermore, we develop a Constraint-Aware Prioritized Experience Replay(CA-PER) buffer that incorporates energy consumption into the TD-error and prioritizes samples near the constraint boundary. This design enhances exploration efficiency and accelerates convergence in constrained domains. Through this unified approach, the proposed framework effectively maximizes video analysis accuracy while minimizing latency and energy consumption, ensuring both system performance and privacy protection in dynamic MEC environments.

Overall, the main contributions of this paper can be summarized as follows:

- We formulate the MEC video task offloading problem as a multi-agent constrained markov game to jointly capture latency, energy, and privacy trade-offs under dynamic network conditions.
- We propose a CPO-MADDPG algorithm equipped with a dual-channel policy gradient, a two-headed critic, and a CA-PER for adaptive and efficient constrained optimization.
- Extensive experiments demonstrate that our method significantly improves video task offloading performance by reducing latency and energy violations while enhancing overall user QoS.

The rest of this article is organized as follows. The relevant approaches in edge computing and fog computing environments are summarized in Section 2. Section 3 introduces the system model and problem formulation. Section 4 describes our DRL-based algorithm model in detail. Section 5 evaluates the performance of the benchmark algorithm we selected in different dimensions. Finally, Section 6 summarizes the full text and provides an outlook for future work.

2. Related work

In this section, we review related work from two main perspectives: privacy-aware offloading with secure resource allocation and multi-objective optimization under resource constraints.

2.1. Privacy-aware offloading and secure resource allocation

With the widespread adoption of edge-assisted video analytics in domains such as autonomous driving, smart surveillance, and intelligent healthcare, privacy leakage has become a central issue in task offloading[17,18]. Unlike traditional cloud computing, MEC performs data processing closer to end devices, reducing latency but also exposing users to potential privacy risks. Offloading video data to untrusted or semi-trusted edge servers enables adversaries to infer sensitive information, such as user location, activity patterns, or device identity, from transmission characteristics[19]. Consequently, designing offloading schemes that preserve privacy while maintaining efficiency has become a pressing research challenge.

To address this, researchers have proposed privacy-preserving mechanisms integrated with deep reinforcement learning (DRL) or optimization frameworks. Li et al. [20] introduced a differential privacy-enhanced DRL algorithm for vehicular edge computing , where Gaussian noise was injected into gradient updates and request data were compressed via autoencoders, achieving dual-layer protection. Zheng et al. [21] developed MACOP, a distributed collaborative offloading framework that jointly considers latency, energy, and privacy. Their method formulated the optimization as a mixed-integer nonlinear programming problem, adopting discretization to balance privacy and performance.

Beyond gradient-based mechanisms, perturbation and encryption approaches have also been explored. Wang et al. [22] proposed CPP-LOM, a collaborative offloading model that uses Laplace perturbation to protect user location and offloading ratios. Jiang et al. [23] introduced a blockchain-based video sharing framework that integrates attribute-based encryption and smart contracts to ensure secure access and tamper-proof transmission.

More recent works aim to jointly optimize privacy and performance within learning frameworks. Sajnani et al. [24] incorporated lightweight AES encryption into DRL-based offloading, explicitly accounting for encryption overhead in energy–latency trade-offs. Fei et al. [25] proposed DRWB, a DRL-based framework that reduces transmission delay and protects sensitive medical data in federated healthcare scenarios.

Although these efforts have advanced privacy-aware offloading, existing methods often rely on static privacy budgets or fixed perturbation levels. Excessive privacy noise degrades model performance, whereas weak protection exposes user data to potential inference attacks. Moreover, these methods rarely consider the coupling between privacy and resource limitations, making it difficult to achieve adaptive privacy protection in dynamic MEC environments with fluctuating channel and energy conditions. This motivates the need for an integrated learning framework that adapts privacy preservation to resource constraints while maintaining system-level performance.

2.2. Multi-objective optimization under resource constraints

In addition to privacy protection, MEC systems are inherently constrained by limited computing power, communication bandwidth, and battery energy[26,27]. These resource constraints lead to dynamic fluctuations in task latency, throughput, and Quality of Experience, posing significant challenges for real-time video analytics[28]. Consequently, optimizing offloading policies under energy and latency constraints has become a key research direction.

DRL has been widely adopted to handle dynamic and uncertain MEC environments. Xu et al. [29] proposed a DRL-based computation offloading framework for virtual reality video transmission, reducing delay and improving stability under high-load conditions. Bi et al. [30] designed a two-stage adaptive 360Â° video streaming scheme combining precaching and real-time scheduling to mitigate field-of-view prediction errors. Wang et al. [31] introduced EDTEA, an event-triggered Markov decision process model that jointly optimizes bitrate adaptation, edge transcoding, and client-side processing.

To further enhance adaptability and scalability, multi-agent reinforcement learning and federated DRL approaches have been explored. Mi et al. [32] developed MAOC, a multi-agent optimization algorithm for energy-harvesting devices, minimizing task latency under uncertain wireless and energy conditions. Lotfolahi et al. [33] proposed a dual-decentralized MAPPO framework for NOMA-assisted industrial IoT, achieving energy-efficient multi-task optimization. Similarly, Zhou et al. [34] combined D3QN with an improved equal bandwidth allocation algorithm to jointly optimize user association, bitrate adaptation, and bandwidth allocation under fluctuating workloads.

Despite these advances, most optimization-based methods treat delay, energy, or accuracy as separate objectives and assume static constraints. Few approaches can simultaneously model multi-objective trade-offs and dynamic energy caps, especially when privacy requirements are intertwined with resource limitations. In dynamic MEC systems, this separation results in unstable performance and suboptimal offloading decisions. Hence, there is a growing demand for a unified optimization framework that jointly considers accuracy, latency, energy consumption, and privacy under dynamic resource constraints–precisely the goal addressed by our proposed CPO-MADDPG framework.

3. MEC system model on task offloading

In this section, we construct a MEC system model for video task offloading. We first introduce the system model and task model, then construct the latency model, energy model and security model, finally formulate the problem. The main notations used in this work are summarized in Table 1.

3.1. System model

We consider the MEC video task offloading scenario shown in Fig. 1. Within the coverage area of each base station, there exists a set of $N = \{1, 2, 3, \dots, n, \dots N\}$ devices. Each device is equipped with an Energy-Harvesting Battery, which will randomly receive energy for replenishment at each time slice t. There are different maximum and minimum battery capacities $B^{\max}$ and $B^{\min}$ for different devices n. Likewise, to offload tasks to the server, each device n has minimum and maximum transmission power allocation thresholds denoted by $P^{\max}$ and $P^{\min}$, respectively. Similar to the [35], these devices are equipped with adjustable camera modules to determine how to sample the video stream data, such as frame rate f, resolution r, etc. We consider the task offloading problem of T time steps, expressed as $T = \{1, 2, 3, \dots, t, \dots T\}$, where a device n generates a video stream task V at each time slice t. Each task V can be divided into K consecutive chunks, expressed as $V = \{1, 2, 3, \dots, k, \dots K\}$. At time t, a video task chunk k on device n is denoted as $n_k(t)$.

It is worth mentioning that within the coverage area of each base station, there is an eavesdropper that will randomly eavesdrop on data. When providing low-latency, high-precision video analysis services, the eavesdropper may infer the terminal's location information and usage pattern by analyzing sampling characteristics such as frame rate and resolution, as well as offloading decisions, thereby leading to the risk of privacy information leakage. For each chunk k, the on-device agent must make an offload decision based on its own battery capacity, privacy requirements, and other attributes, choosing whether to process each chunk locally or offload it. Our goal is to ensure task privacy while staying within the energy budget to achieve the best possible user QoS.

3.2. Task model

As illustrated in Fig. 2, Each device n has an agent that can make the best sampling and unloading decisions within a time slice t based on its current remaining energy. For each video offloading task V, the offloading decision of each chunk k at time t is determined by the agent.

Table 1
Summary of main notations.

Notation	Definition
N	Set of user devices
T	Set of time slices
V	Video stream task generated by a UD at each time slice
K	Number of consecutive chunks in a video task,
$B^{\max}$	Maximum battery capacities
$B^{\min}$	Minimum battery capacities
$P^{\max}$	Maximum transmission power allocation thresholds
$P^{\min}$	Minimum transmission power allocation thresholds
$F^{\max}$	Maximum video frame rate
$F^{\min}$	Minimum video frame rate
$R^{\max}$	Maximum video resolution
$R^{\min}$	Minimum video resolution
$C^{\max}$	Maximum allocated CPU operating frequency
$C^{\min}$	Minimum allocated CPU operating frequency
x_{n_k}	Offloading decision of chunk k on device n
r_{n_k}	Sampling resolution of video chunk k on device n
f_{n_k}	Sampling frame rate of video chunk k on device n
c_{n_k}	CPU resources allocated of chunk k on device n
p_{n_k}	Transmission power allocated of chunk k on device n
S_{n_k}	Size of chunk k on device n
ξ	The number of CPU cycles required to process one bit of video
$L^L_{n_k}$	Latency of local processing for chunk k on device n
$L^M_{n_k}$	Latency of MEC processing for chunk k on device n
R_{n_k}	Upload rate of chunk k from device n
W	Channel bandwidth
g_{n_k}	Channel gain of device n
σ^2	Noise power of Gaussian white noise
$E^L_{n_k}$	Local computing energy consumption for chunk k
$E^M_{n_k}$	Energy consumption for offloading chunk k
E_{n_k}	Remaining energy of device n
$E^{up}_{n_k}$	Energy harvested by device n at time t
$D^e_{n_k}$	Eavesdropping rate for chunk k on device n
$D^s_{n_k}$	Secrecy rate of chunk k on device n
V_{n_k}	Video recognition quality of chunk k on device n
A_{n_k}	Average accuracy over time slices for device n
F_{n_k}	Video smoothness over time slices for device n
L_{n_k}	Total latency for chunk k on device n
E_{n_k}	Total energy consumption for chunk k on device n
C_{n_k}	Computational cost of chunk k on device n
P_{n_k}	Penalty when battery consumption exceeds threshold D_n
R_n	Overall Reward of device n

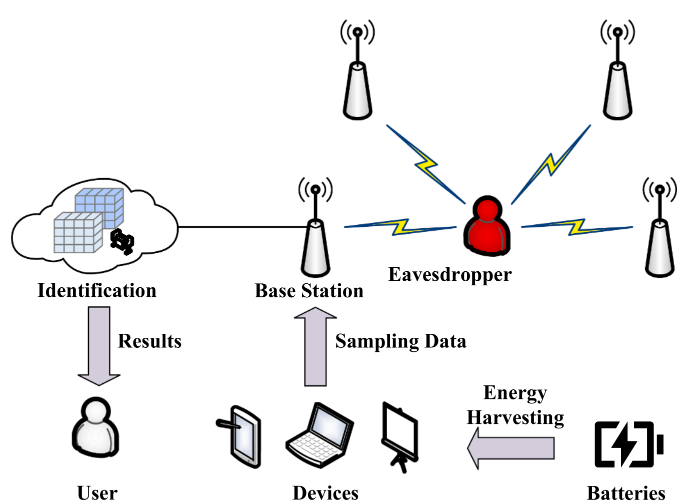

Fig. 1. In a MEC video task offloading scenario, the device is equipped with an energy harvesting battery to randomly replenish energy. For the offloaded video task, there is an eavesdropper in the environment that randomly eavesdrops on all uploaded data within the base station range. After securely receiving the task, the base station performs video recognition and other processing to obtain the results and finally returns them to the user.

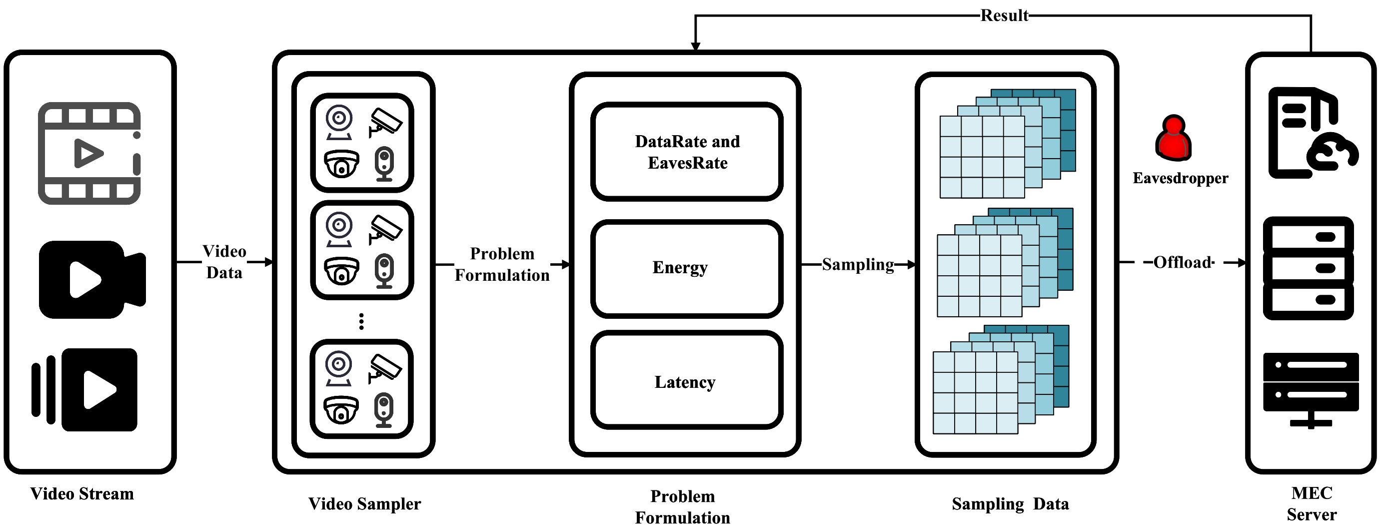

Fig. 2. In the agent decision-making scenario, the agent determines the video sampling configuration based on the current system state and samples the video stream data. Based on the sampled data, the problem is formulated to obtain the delay and energy consumption, and the potential eavesdropping risk is estimated based on the transmission rate. Finally, the agent decides whether to upload the sampled data to the edge server via the wireless link for offloading.

The video sampler on the device performs raw sampling of the video stream data, determines the frame rate and resolution of the chunk k in each time slice t, and formulates the problem based on the sampled chunk data to make offloading decisions. Finally, the chunk data is uploaded to the mec server for offloading. During the offloading process, if the upload rate is less than the eavesdropping rate, the task is not allowed to be offloaded, and the sampled video chunks are calculated locally. This will be described in detail in the Section 3.5.

Each chunk k on the device n is treated as an independent task, and the agent's offloading decision is determined by five decision variables: a binary variable of whether to offload the processing task x_{n_k} , video stream sampling resolution r_{n_k}, video stream sampling frame rate f_{n_k}, the locally allocated CPU computing resources c_{n_k}, and transmission power p_{n_k}. For each chunk k on device n, we describe whether to offload by using a binary variable of whether to offload the processing task, which can be expressed as follows:

$$x_{n_k} = \begin{cases} 0, & Local processing \\ 1, & MEC processing \end{cases} . \tag{1}$$

The agent needs to make the best sampling and unloading decisions based on the device attributes. Similar to [36], the size of chunk k is determined by the sampling decision, which can be expressed as follows:

$$S_{n_k}(t) = \alpha f_{n_k}(t) {r_{n_k}}^2(t), \tag{2}$$

where the parameter α represents the number of bits carried by a pixel, $f_{n_k}(t)$ and $r_{n_k}(t)$ represent the frame rate and resolution sampled by device n at time slice t, respectively.

3.3. Latency model

When processed locally, the latency of the video task chunk k at time slice t is mainly caused by the device's processing of the task. The local video chunk k processing latency is determined by the locally allocated computing resources and can be expressed as:

$$L^L_{n_k}(t) = (1 - x_{n_k}(t)) \frac{S_{n_k}(t)\xi}{c_{n_k}}, \tag{3}$$

where c_{n_k} represents the CPU operating frequency assigned to agent, and parameter ξ is the number of CPU cycles required to process one bit of video task.

When the task is offloaded to the mec server, the processing delay is mainly composed of the upload delay and the mec server processing delay, expressed as:

$$L^M_{n_k}(t) = x_{n_k}(t)(\frac{S_{n_k}(t)}{D_{n_k}(t)} + \frac{S_{n_k}(t)\xi}{MEC_c}), \tag{4}$$

where the former represents the transmission delay of chunk k, the latter represents the processing delay of resource-intensive MEC server, MEC_c represents the fixed computing resources of MEC server, and $D_{n_k}(t)$ represents the upload rate of the chunk k, expressed as:

$$D_{n_k}(t) = W \log_2 \left(1 + \frac{p_{n_k}(t) g_{n_k}(t)}{\sigma^2 + \sum_{\substack{n'=1 \\ n' \neq n}}^{N} x_{n'_k}(t) p_{n'_k}(t) g_{n'_k}(t)} \right), \tag{5}$$

where W represents the channel bandwidth, $p_{n_k}(t)$ represents the transmit power, $g_{n_k}(t)$ represents the channel gain, and σ represents the noise density of Gaussian white noise.

3.4. Energy model

Similar to the [37]. Initially, each device n is full with its maximum battery capacity of bmax n. The battery level in the next time interval depends on the energy consumed and harvested.

The energy consumed locally to process chunk k at time t is expressed as:

$$E^L_{n_k}(t) = (1 - x_{n_k}(t) \varsigma c^2_{n_k}(t) S_{n_k}(t) \xi, \tag{6}$$

where ς is the energy coefficient, which varies with different physical devices. The energy cost of offloading tasks from user n to a resource-intensive server is highly correlated with the amount of offloaded tasks and the transmission rate of the terminal device, which is expressed as:

$$E^M_{n_k}(t) = x_{n_k}(t) p_{n_k}(t) \frac{S_{n_k}(t)}{D_{n_k}(t)}. \tag{7}$$

So the remaining energy of device ud at time t can be expressed as:

$$E^r_{n_k}(t+1) = max(E_{n_k}(t) - E^M_{n_k}(t) - E^L_{n_k}(t) + E^{up}_{n_k}), B^{max}), \tag{8}$$

where $E^{up}_{n_k}$ represents the energy that the device receives from the energy harvesting battery at time t, $B^{\max}$ represents the maximum battery capacities.

3.5. Security model

Without loss of generality, we assume a global passive eavesdropper that can listen to all uplink transmissions. Although its physical location is not explicitly modeled, its effective eavesdropping capability is characterized by the channel gain $g^e_{n_k}(t)$, whose magnitude is varied in simulations to emulate different attack strengths (e.g., near/far or favorable/adverse conditions). Similar to [16], we denote the eavesdropping rate as:

$$D^e_{n_k}(t) = W\log_2\left(1+\frac{p^e(t)g^e_{n_k}(t)}{\sigma_e^2+\sum_{\substack{n'=1\\ n'\neq n}}^{N} x_{n'_k}(t)p_{n'_k}(t)g^e_{n'_k}(t)}\right), \tag{9}$$

where σ_e represents the eavesdropper's Gaussian noise, $p^e(t)$ represents the eavesdropper's transmission power. In our simulation, the eavesdropper channel gain is modeled as a random variable to abstractly characterize different attack strengths. Specifically, the true eavesdropper channel gain is generated as

$$g_{n_k}(t) \sim \mathcal{U}[g_{\min}, g_{\max}], \tag{10}$$

where the interval $[g_{\min}, g_{\max}]$ controls the overall eavesdropping strength. Larger values correspond to stronger eavesdropping capability. Due to imperfect channel state information, only an estimated gain $g^e_{n_k}(t)$ is available, which satisfies the bounded-error model:

$$g^e_{n_k}(t) \in G(t) = \left\{ |g_{n_k}(t) - g^e_{n_k}(t)| \le \rho \right\}, \tag{11}$$

where ρ denotes the maximum estimation error bound, we set $\rho = 0.1g_{\max}$, G represents the set of all estimated channel gains at time t. In simulations, we sample $g^e_{n_k}(t)$ uniformly from the interval $[g_{n_k}(t) - \rho,\ g_{n_k}(t)+\rho]$. Based on the obtained $D^e_{n_k}(t)$ and $D_{n_k}(t)$, the secrecy rate in the worst-case can be obtained by:

$$D^s_{n_k}(t) = D_{n_k}(t) - W\log_2\left(1+\frac{p^e(t)(g^e_{n_k}(t))^{max}}{\sigma_e^2+\sum_{\substack{n'=1\\ n'\neq n}}^{N} x_{n'_k}(t)p_{n'_k}(t)(g^e_{n'_k}(t))^{min}}\right) \tag{12}$$

where $(g^e_{n_k}(t))^{\max}$ and $(g^e_{n_k}(t))^{\min}$ represent the maximum and minimum values in the set $G(t)$, respectively. For the task chunk k, transmission security can be achieved only when the secrecy rate is positive; thus, the security model can be expressed by:

$$D^s_{n_k}(t) > 0. \tag{13}$$

In our system, task offloading is allowed only when the secrecy condition is satisfied. Specifically, if an agent selects an offloading action $x_{n_k} = 1$ that results in $D^s_{n_k}(t) \le 0$, the offloading request is regarded as insecure and adjusting the offloading action into $x_{n_k} = 0$. In this case, the task is executed locally, ensuring that all executed offloading decisions satisfy the secrecy constraint.

3.6. Problem formulation

This paper takes video recognition quality $V_{n_k}(t)$ and computational cost $C_{n_k}(t)$ as the core indicators of video security offloading and models them uniformly as user QoS. Video recognition quality is jointly concerned with accuracy and smoothness, while computational cost is a combination of latency and energy consumption. Specifically, the video recognition quality of the time slice t is defined as the weighted sum of accuracy and fluency, which is defined as:

$$V_{n_k}(t) = \mu_1 A_{n_k}(t) + \mu_2 F_{n_k}(t), \mu_1+\mu_2 = 1, \tag{14}$$

where $A_{n_k}(t)$ is the accuracy rate at t time slice is expressed as the average of the accuracy rates in the previous t time slices, expressed as:

$$A_{n_k}(t) = \frac{1}{t}\sum_{t'=1}^{t} a_{n_k}(t') \tag{15}$$

According to [36], the parsing accuracy $a_{n_k}(t)$ of the video chunk k at time t expressed as:

$$a_{n_k}(t) = 0.988 - 4.469e^{\frac{-r_{n_k}(t)}{200}}. \tag{16}$$

Similarly, we define video smoothness $F_{n_k}(t)$ as the quantitative representation of frame smoothness and frame change rate over the past t time slices, defined as:

$$F_{n_k}(t) = \frac{1}{t}\sum_{t'=1}^{t}\left(f^{fs}_{n_k}(t')f^{fc}_{n_k}(t')\right) \tag{17}$$

$$f^{fs}_{n_k}(t) = \frac{f_{n_k}(t)}{f_{max}}, \tag{18}$$

$$f^{fc}_{n_k}(t) = (1-\frac{|f_{n_k}(t)-f_{n_k}(t-1)|}{f_{n_k}(t-1)})), \tag{19}$$

where the frame smoothness $f^{fs}_{n_k}(t)$ is related to the current frame rate $f_{n_k}(t)$ and the maximum frame rate $f_{\max}$, the frame rate of change $f^{fc}_{n_k}(t)$ is expressed as the degree of change in the frame rate over the first T time slices.

Similarly, we define the computational cost as the sum of latency and energy consumption, expressed as:

$$C_{n_k}(t) = \beta_1 L_{n_k}(t) + \beta_2 E_{n_k}(t), \beta_1+\beta_2 = 1, \tag{20}$$

where $L_{n_k}(t)$ is the sum of the processing latencies, determined by formulas (3) and (4), and expressed as:

$$L_{n_k}(t) = L^L_{n_k} + L^M_{n_k}. \tag{21}$$

Similarly, $E_{n_k}(t)$ is the sum of energy consumption, which is determined by formulas (6) and (7), expressed as:

$$E_{n_k}(t) = E^L_{n_k} + E^M_{n_k}. \tag{22}$$

To ensure safe video offloading under resource-constrained conditions, we use the portion exceeding the battery threshold as a penalty, denoted as:

$$P_{n_k}(t) = max(BT - E^r_{n_k}(t), 0) \tag{23}$$

where BT represents the battery threshold of device n,the default value is 0.3 of $B^{\max}$. The overall QoS of the user is determined by the video recognition quality and the computational cost, which is expressed as:

$$R_n(t) = \sum_{k=1}^{K}(V_{n_k}(t) - C_{n_k}(t) - P_{n_k}(t)). \tag{24}$$

Based on the multi-constrained video task offloading problem, our optimization goal is to maximize user QoS while satisfying system constraints. This is expressed as:

$$maximize \sum_{t=1}^{T}\sum_{n=1}^{N} R_n(t), \tag{25}$$

$$s.t. C_1 : x_{n_k} \in \{0,1\}, \forall n \in N, \tag{26}$$

$$C2 : P^{min} \le p_{n_k} \le P^{max}, \forall n \in N, \tag{27}$$

$$C3 : C^{min} \le c_{n_k} \le C^{max}, \forall n \in N, \tag{28}$$

$$C4 : R^{min} \le r_{n_k} \le R^{max}, \forall n \in N, \tag{29}$$

$$C5 : F^{min} \le f_{n_k} \le F^{max}, \forall n \in N, \tag{30}$$

$$C6 : D^s_{n_k}(t) > 0, \forall n \in N, \tag{31}$$

where Eq. (26) implies that a chunk task is processed locally or uploaded to the MEC server, Eq. (27) and Eq. (28) represents the threshold ranges for transmit power and allocated CPU operating frequency, respectively. Similarly, Eq. (29) and (30) represent the frame rate and resolution sampling interval of the video block, respectively. And Eq. (31) represent the transmission security can be achieved only when the secrecy rate is positive.

Our goal is that the intelligent agent can optimize the offloading decision for different tasks under different wireless link load conditions, and complete the intelligent agent through local offloading decision to find an optimal computing resource allocation, or the offloading decision to find the optimal power selection and reasonable wireless link selection to maximize the user QoS.

4. Markov decision and algorithm

In this section we will introduce markov decision based on the above mec system model and the DRL algorithm we designed.

4.1. Markov decision

4.1.1. State

In our MEC environment, some constant states such as server computing power MEC_c, gaussian noise σ and σ_e, battery threshold BT are excluded from the state space. The state of device n at time t for video chunk k is represented as:

$$S_{n_k}(t) = \{P^{max}, P^{min}, C^{max}, C^{min}, g_{n_k}(t), E_{n_k}(t), E^{up}_{n_k}, b_{n_k}(t)\}, \tag{32}$$

where $P^{\max}$ and $P^{\min}$ are the maximum transmit power and the minimum transmit power of the device n, $C^{\max}$ and $C^{\min}$ are the maximum allocated CPU operating frequency and the minimum allocated CPU operating frequency of the device n, $g_{n_k}(t)$ is the channel gain of the device n, $E_{n_k}(t)$ is the remaining disposable energy of the device n at time t, $E^{up}_{n_k}$ is the energy replenished of the device n at time t, b_{n_k} is the channel bandwidth of the device n at time t. The state space composed of all devices at time t is:

$$S_n(t) = \{S_{n_k}(t), n \in N, k \in K, t \in T\}. \tag{33}$$

4.1.2. Action

At the beginning of each time step, the device's agent make a decision action regarding its resource allocation, represented as:

$$A_{n_k}(t) = \{x_{n_k}(t), f_{n_k}(t), r_{n_k}(t), p_{n_k}(t), c_{n_k}(t)\}, \tag{34}$$

where $x_{n_k}(t)$ implies that a chunk task is processed locally or uploaded to the MEC server, $f_{n_k}(t)$ and $, r_{n_k}(t)$ represent the video block sampling frame rate and resolution selected by the agent, $p_{n_k}(t)$ represent the transmission power of the chunk task offloaded to the MEC server, $c_{n_k}(t)$ represents the computing resources allocated when the chunk task is processed on the local device. The action space composed of all devices at time t is:

$$A_n(t) = \{A_{n_k}(t), n \in N, k \in K, t \in T\}. \tag{35}$$

4.1.3. Reward

Our optimization objective is to maximize the team reward function of the multi-agent system, as shown in Eq. 25, we define the reward as the total QoS of all users in a period of time. The environment returns the reward immediately after all agents take actions in each period of time, which can be expressed as:

$$R(S_n(t), A_n(t)) = R_n(t). \tag{36}$$

4.2. Algorithm implementation

4.2.1. Constrained markov game

In our model, our goal is distributed constraint equilibrium, and the agents belong to centralized training/distributed execution. Specifically, each mobile device is modeled as an independent agent, capable of making independent local offloading decisions under its own distinct constraints. The interaction experiences of all agents, including state, action, cost, and next state, are stored in a shared replay buffer to train each agentâ's neural networks and approximate the mapping from state–action pairs to Q-values. We formalize the MEC video offloading problem as a CMG $\mathcal{G} = \langle S, A, P, R, E\rangle$. Where S represents the set of states consisting of all agents, denoted as $S = \{S_1(t), S_2(t) \dots S_n(t)\}$.

Algorithm 1: Constrained Policy Optimization Multi-Agent DDPG (CPO-MADDPG).

Input : Number of agents N, maximum episodes E, steps per episode T, replay buffer size N_{buffer}, learning rates $\alpha_\pi, \alpha_Q, \eta$, discount factors γ_r, γ_c, constraint limit C_{limit}
Output: Trained actor π_{θ^π} and two-headed critic networks Q_r, Q_c

```
Initialize: actor π_θπ and target actor π_θπ^target; dual-critic Q_r, Q_c
    and target critics; replay buffer D; λ ← 0;
for episode = 1 to E do
    Initialize environment state s_0;
    for step t = 0 to T − 1 do
        foreach agent i do
            Select action with exploration noise:
              a_t^i = π_θπ(s_t^i) + N_t;
        Execute actions a_t = {a_t^i}_{i=1}^N and observe next state s_{t+1},
          reward r_t, cost c_t;
        Store (s_t, a_t, r_t, c_t, s_{t+1}) into D;
        CA-PER : Compute TD errors
          σ^r = r_t + γ_r Q_r^target(s_{t+1}, a_{t+1}) − Q_r(s_t, a_t);
        Compute TD errors
          σ^c = c_t + γ_c Q_c^target(s_{t+1}, a_{t+1}) − Q_c(s_t, a_t);
        Compute priority p_i = (|σ^r| + |σ^c| + ε)^α;
        Sample mini-batch from D according to p_i with
          importance weight w_i;
        Update two-header Critic: Minimize
          L_critic = E[(Q_r − y_r)^2 + (Q_c − y_c)^2];
        Update Actor (Dual-Channel Policy Gradient):
          ∇_θπ L = −∇_θπ J_r + λ∇_θπ J_c;
        θ^π ← θ^π − α_π ∇_θπ L;
        Update Lagrange Multiplier:
          λ ← max(0, λ + η(J_c(π) − C_limit));
        Update Target Networks:
          θ^{π^target} ← τθ^π + (1 − τ)θ^{π^target};
        θ^{Q^target} ← τθ^Q + (1 − τ)θ^{Q^target};
        s_t ← s_{t+1};
return π_θπ, Q_r, Q_c;
```

Similarly, A represents the set of actions chosen by all agents, denoted as $A = \{A_1(t), A_2(t) \dots A_n(t)\}$. The reward R is calculated as shown in Eq. (25), and our objective function is also shown in Eq. (26). The E represents the remaining energy of the current device, determined by the formula Eq. (8), the constraint C_{limit} for each agent is defined as $0.05B^{\max}$. At time step t, each agent makes action choices $A_n(t) \in A$ based on local observations $S_n(t) \in S$. The global state S collects all devices' local observations. The environment transitions according to $P(S_n(t+1) \mid S_n(t), A_n(t)$.

4.2.2. CPO-MADDPG

Under bounded action space and energy budget, while a conservative always-local strategy for resource allocation can meet cost constraints, it can lead to increased computational costs, thereby affecting user QoS. To solve the user QoS maximization problem in Eq. (25) and seek the optimal constrained strategy for the team, we design a DRL-based CPO-MADDPG Algorithm 20, whose overall architecture is illustrated in Fig. 3.

Traditional DRL-based MDPs often employ single-objective optimization, with maximizing cumulative rewards as the sole learning objective, expressed as:

$$maximize J_r(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{T} r(s_t, a_t)\right], \tag{37}$$

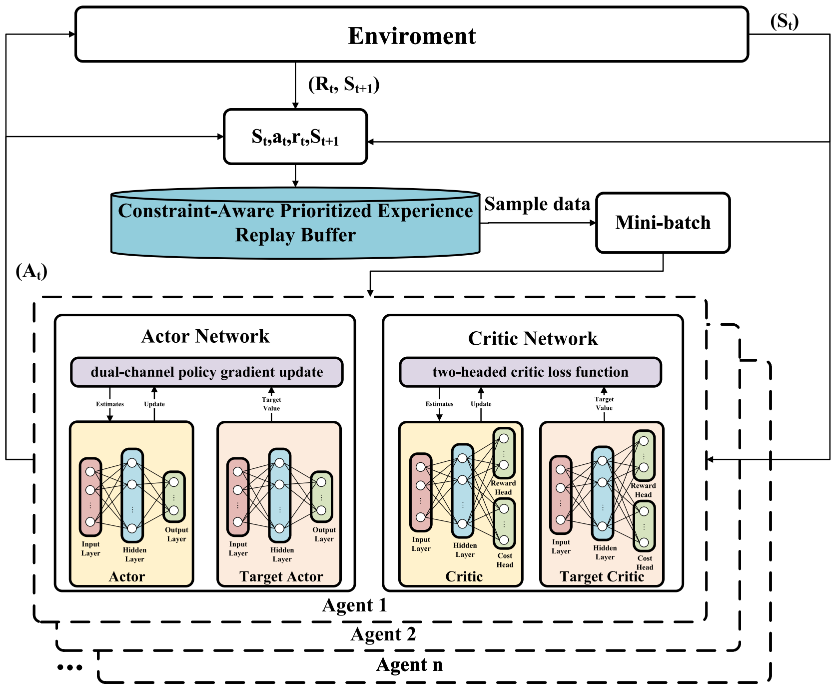

Fig. 3. Overall architecture of the proposed CPO-MADDPG framework for MEC task offloading. The system consists of multiple agents, each composed of an Actor–Critic pair with corresponding target networks. At each time step t, agents observe the system state S_t and take offloading actions A_t according to the actor network. The environment returns the next state S_{t+1} and corresponding reward–cost pair (R_t, C_t), which are stored in the CPO Experience Replay buffer. Mini-batches sampled from the buffer are used to update both the actor and critic networks through the CPO optimizer and loss function, respectively. The critic network adopts a dual-head structure to estimate both the reward value and constraint cost, enabling adaptive policy learning that simultaneously maximizes task performance while satisfying energy and privacy constraints across all agents.

where $r(s_t, a_t)$ represents the calculation of reward for action a_t in state s_t, $\mathbb{E}_\pi$ represents the expectation generated under strategy π.

Unlike conventional DRL algorithms that optimize a single objective (usually cumulative reward), MEC task offloading inherently involves multi-objective and constrained optimization, where energy consumption and privacy leakage must be explicitly bounded. Traditional penalty-based methods, which integrate constraint violations as penalty terms within the reward, often face a parameter sensitivity issue: excessive penalty weights overly suppress reward learning, while insufficient ones cause constraint violations. To overcome this dilemma, we formulate the problem as a CMG, where reward maximization and constraint satisfaction are treated as separate yet coupled objectives. Formally, the optimization problem is given as:

$$maximize J_r(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{T} r(s_t, a_t)\right], \tag{38}$$

$$s.t. J_c(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{T} c(s_t, a_t)\right] \le C_{limit}, \tag{39}$$

where $J_r(\pi)$ denotes the cumulative reward, $J_c(\pi)$ represents the total energy consumption bounded by the constraint C_{limit}, $r(s_t, a_t)$ represents the energy cost function of action a_t in state s_t.

4.2.3. Lagrange multiplier method

To solve this constrained optimization problem, we employ Lagrangian relaxation, transforming it into an unconstrained form:

$$L(\pi, \lambda) = J_r(\pi) - \lambda(J_c(\pi) - C_{limit}), \lambda \ge 0, \tag{40}$$

where λ is updated dynamically during training:

$$\lambda \leftarrow \max(0, \lambda + \eta(J_c(\pi) - C_{limit})), \tag{41}$$

where η is the learning rate. This update rule ensures that λ gradually increases when energy consumption exceeds the limit, and remains constant otherwise, thereby gradually converging to a solution that satisfies the constraints during the optimization process,ensuring the policy gradually converges to a feasible solution that satisfies the energy constraint.

4.2.4. Policy gradient

Based on constraint modeling, this paper further proposes a dual-channel policy gradient update mechanism. Traditionally, the DDPG policy gradient update mechanism and the Actor gradient update can be expressed as:

$$L_{actor} = -J_r(\pi). \tag{42}$$

$$\nabla_{\theta^\pi} L(\pi) = -\nabla_{\theta^\pi} J_r(\pi), \tag{43}$$

where ∇_{θ^π} represents the gradient of all parameters under the policy π .

This single gradient update method cannot directly describe the optimization direction of energy consumption constraints, which often leads to the policy bias towards maximizing short-term rewards while ignoring long-term energy efficiency. To solve this problem, this paper proposes a dual-channel policy gradient, which updates policies based solely on a reward-value function, this actor update utilizes both reward and cost critic information to achieve joint optimization of both revenue and energy constraints. Specifically, the actor's optimization objective is defined as:

$$L_{actor} = -J_r(\pi) + \lambda_{\max}(0, J_c(\pi) - C_{limit}). \tag{44}$$

Based on this optimization goal, the gradient update of the Actor can be expressed as:

$$\nabla_{\theta^\pi} L(\pi) = -\nabla_{\theta^\pi} J_r(\pi) + \lambda \nabla_{\theta^\pi} J_c(\pi), \tag{45}$$

which allows the policy to balance reward improvement and constraint satisfaction adaptively, without the need for second-order updates required by traditional CPO methods, achieving approximate constraint satisfaction with low computational complexity. Since the actor in CPO-MADDPG follows a deterministic policy, the policy gradients for both reward and cost objectives are computed using the deterministic policy gradient theorem. Specifically, the gradients are obtained by backpropagating through the corresponding critic networks. For the reward channel, the gradient is estimated as:

$$\nabla_{\theta^\pi} J_r(\pi) \approx \mathbb{E}_{s\sim D}\left[\nabla_a Q_r(s,a)\Big|_{a=\pi(s)} \nabla_{\theta^\pi} \pi(s)\right], \tag{46}$$

and similarly, the cost gradient is computed via the cost critic as

$$\nabla_{\theta^\pi} J_c(\pi) \approx \mathbb{E}_{s\sim D}\left[\nabla_a Q_c(s,a)\Big|_{a=\pi(s)} \nabla_{\theta^\pi} \pi(s)\right], \tag{47}$$

where D denotes the replay buffer. To prevent scale dominance between reward and cost gradients, we normalize the cost gradient by the ratio of gradient norms:

$$g_r = \nabla_{\theta^\pi} J_r, \quad g_c = \nabla_{\theta^\pi} J_c, \quad \tilde{g}_c = g_c \cdot \frac{\|g_r\|_2}{\|g_c\|_2 + \epsilon}. \tag{48}$$

The final actor update direction is given by $-g_r + \lambda \tilde{g}_c$. In addition, gradient clipping with a maximum norm of 1.0 is applied to further enhance training stability. No advantage estimator or baseline is employed, since the deterministic policy gradient directly propagates gradients through the critic networks.

4.2.5. Two-headed critic

To support the dual-channel policy gradient, we further propose a two-headed critic network and adopt an end-to-end training approach that shares a common feature extractor but diverges into two output branches: a reward channel $Q_r(s,a)$: used to estimate reward return; a cost channel $Q_c(s,a)$: used to estimate energy consumption. In the two-headed critic network, reward estimation and constraint estimation are decoupled, but the underlying representation is shared, ensuring both training stability and constraint interpretability. It can be expressed as:

$$Q_r(s,a) = r(s,a) + \gamma_r Q_r^{target}(s',a'), \tag{49}$$

$$Q_c(s,a) = c(s,a) + \gamma_c Q_c^{target}(s',a'), \tag{50}$$

where γ_r and γ_c are the discount factors for reward and cost, respectively. The two-headed critic loss function is:

$$L_{critic} = \mathbb{E}\left[(Q_r(s,a) - y_r)^2 + (Q_c(s,a) - y_c)^2\right]. \tag{51}$$

This structure enables simultaneous estimation of task rewards and constraint costs within a unified network, ensuring efficient training and improving the interpretability of constraint enforcement. We employ different discount factors for the two heads since they correspond to objectives with different temporal characteristics: the reward signal reflects long-term QoS optimization, while the cost signal captures short-term energy and security violations that require faster correction. Empirically, using $\gamma_c < \gamma_r$ improves constraint responsiveness without sacrificing long-term performance.

To mitigate negative interference between reward and cost estimation, the critic adopts a shared feature extractor followed by two decoupled output heads. This design allows low-level representations of system dynamics to be shared, while preventing conflicting gradients from directly affecting the task-specific outputs.

We maintain one target critic with the same shared-backbone two-head structure. The reward head and cost head use their corresponding target outputs Q_r and Q_c with synchronized soft updates, which further stabilizes training. In our implementation, the critic network consists of a three-layer fully connected backbone, followed by two linear output heads for Q_r and Q_c, respectively. The shared backbone consists of three fully-connected layers with 256, 128, and 64 neurons, respectively, each followed by ReLU activation. Both reward and cost heads are implemented as one linear layer with 64 hidden units and a linear output.

4.2.6. CA-PER

To improve sample efficiency and accelerate policy convergence under constraints, we propose the CA-PER buffer. Building on the traditional prioritized experience replay mechanism, this buffer incorporates both reward TD error and constraint TD error into the priority computation, enabling joint priority sampling for both objectives. In this way, transitions that are critical for either performance improvement or constraint satisfaction are replayed more frequently.

Table 2
Parameters used in the simulations.

Param	value	param	value
N	20	episode	2000
actor learning	0.001	critic learning	0.001
k	5×10^{-28}	ρ	$0.1g_{\max}$
Bandwidth	120MHZ	α	12
μ_1,μ_2	0.5,0.5	β_1,β_2	0.5,0.5
η	0.001	$\lambda_{\max}$	1
$B^{\max}$	8	C_{limit}	$0.05D_n$
$g_{\min}$	8db	$g_{\max}$	10db

Specifically, for each transition i, we compute the reward TD error σ_i^r and the cost TD error σ_i^c. Since these two TD errors may have heterogeneous scales, we first normalize them within each mini-batch:

$$\hat{\sigma}_i^r = \frac{\sigma_i^r - \mu_r}{s_r + \delta}, \quad \hat{\sigma}_i^c = \frac{\sigma_i^c - \mu_c}{s_c + \delta}, \tag{52}$$

where μ_r, μ_c and s_r, s_c denote the batch mean and standard deviation of reward and cost TD errors, respectively, and δ is a small constant to avoid numerical instability. The combined priority is then defined as:

$$p_i = (|\hat{\sigma}_i^r| + |\hat{\sigma}_i^c| + \varepsilon)^\alpha, \tag{53}$$

where α controls the degree of prioritization and ε is a small constant ensuring that all samples have non-zero probability to be replayed. The sampling probability of transition i is given by:

$$P(i) = \frac{p_i}{\sum_j p_j}. \tag{54}$$

Since prioritized sampling introduces bias, we adopt importance-sampling weights to correct this bias:

$$w_i = \left(\frac{1}{N \cdot P(i)}\right)^{\beta_t}, \quad \tilde{w}_i = \frac{w_i}{\max_j w_j}, \tag{55}$$

where N is the replay buffer size, and β_t is annealed from β_0 to 1.0 during training to gradually increase the strength of bias correction. In all experiments, we set $\alpha = 0.6$, $\beta_0 = 0.4$, and linearly anneal β_t to 1.0 over the entire training process. We use $\varepsilon = 10^{-6}$ and $\delta = 10^{-6}$. During training, the normalized improve sample weight $\tilde{w}_i$ is applied to weight the critic losses, i.e.,

$$L = \mathbb{E}_i\left[\tilde{w}_i\left((Q_r(s_i,a_i) - y_i^r)^2 + (Q_c(s_i,a_i) - y_i^c)^2\right)\right], \tag{56}$$

which effectively reduces the bias introduced by prioritized sampling and stabilizes training.

4.3. Complexity analysis

The complexity of the proposed CPO-MADDPG algorithm primarily stems from the multi-agent setup and the use of two-headed critics with CA-PER. The per-agent complexity for action selection and gradient update is $O(N)$. But the complexity of the two-headed critic structure and the CA-PER buffer introduces a minor additional overhead, making the overall algorithm complexity higher than traditional methods like MADDPG or MAPPO, which have a complexity of $O(N)$ for centralized training. Despite the increased complexity, the performance improvements in terms of task offloading efficiency, privacy protection, and energy consumption make the additional computational cost justifiable.

5. Simulation experiment

5.1. Implementation

We conducted extensive simulation experiments to evaluate our proposed CPO-MADDPG algorithm in a MEC environment, where a single base station serves multiple users. The users are randomly distributed around the base station, and each mobile device is modeled as an independent agent making local offloading decisions under its own distinct constraints. The key parameters used in the simulations are summarized in Table 2. Specifically, the number of users, denoted as N, is set to 20, and the system operates for a total of 2000 training episodes. The bandwidth available for communication is 120 MHz, with noise power considered under typical communication conditions. The learning rate for both the actor and critic networks is set to 0.001, ensuring a balance between convergence speed and stability. The learning rate η of the Lagrange multipliers λ, which controls the trade-off between reward maximization and constraint satisfaction, is selected with 0.001 based on preliminary tests. To prevent instability during training, we clip the value of the Lagrange multiplier λ to a maximum of 1. For the energy constraints, the maximum battery capacity of device $B^{\max}$ is set to 8 units, and the maximum allowable energy consumption, C_{limit}, is defined as 5% of the $B^{\max}$. The parameter ρ, which defines the uncertainty in the eavesdropper's channel estimation, is set to 0.1 times the maximum channel gain, $g_{\max}$. This set of parameters was chosen to reflect a realistic MEC scenario with multiple users competing for shared resources, ensuring a comprehensive evaluation of the proposed algorithm's performance. To evaluate the advantages of our proposed algorithm, we compared our proposed deep reinforcement learning approach with several benchmark heuristics. In our implementation, we maintain one target critic with the same shared-backbone two-head structure. The reward head and cost head use their corresponding target outputs Q_r and Q_c with synchronized soft updates, which further stabilizes training. The critic network consists of a three-layer fully connected backbone, followed by two linear output heads for Q_r and Q_c, respectively. The shared backbone consists of three fully connected layers with 256, 128, and 64 neurons, each followed by ReLU activation. Both reward and cost heads are implemented as one linear layer with 64 hidden units and a linear output. For the actor network, we use a fully connected neural network with three hidden layers. The input layer size corresponds to the state space dimension, and the hidden layers consist of 256, 128, and 64 neurons, each followed by ReLU activation. The output layer generates the offloading action, with a dimension corresponding to the action space, and uses a Tanh activation function to ensure that the actions lie within the valid range. The benchmark experiments used for comparison are as follows:

5.1.1. CPO-MADDPG

We propose a algorithm based on DRL, which integrates constrained optimization theory into the traditional DDPG framework. The algorithm employs a two-headed critic structure and a dual-channel policy gradient mechanism to jointly optimize the long-term reward and energy consumption constraints. Additionally, CA-PER mechanism is introduced to enhance learning efficiency and ensure constraint satisfaction during training.

5.1.2. Multi-agent deep deterministic policy gradien(MADDPG)

MADDPG is a typical multi-agent DRL algorithm that extends the traditional DDPG framework to cooperative and competitive environments. Each agent maintains its own actor-critic pair. During training, the centralized critic network receives the joint observations and actions of all agents to accurately evaluate Q-values, while each actor only uses its own local observation for decentralized execution.

5.1.3. Lagrange multi-agent deep deterministic policy gradien(LMADDPG)

A constraint reinforcement learning algorithm that combines Lagrangian constraints with the MADDPG algorithm.

5.1.4. Multi-agent proximal policy optimization(MAPPO)

MAPPO is a widely used on-policy DRL algorithm that stabilizes training by clipping the policy update ratio. It achieves a balance between sample efficiency and stability by preventing large policy deviations.

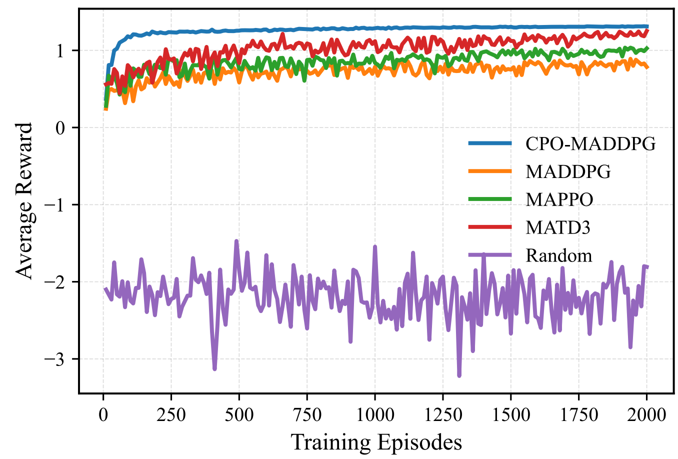

Fig. 4. Convergence of average reward.

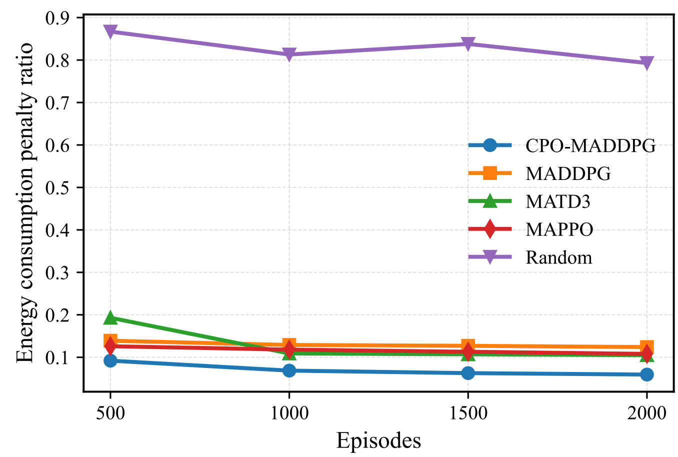

Fig. 5. Energy Consumption Comparison.

5.1.5. Multi-agent twin-Delayed deep deterministic policy gradient (MATD3)

MATD3 enhances the traditional DDPG framework by introducing double critics, target policy smoothing, and delayed policy updates to mitigate overestimation bias and improve training stability, serving as a strong baseline for continuous control tasks.

5.1.6. Random

As a non-learning baseline, the random policy assigns offloading actions randomly. Each task is either executed locally or offloaded to the MEC server with equal probability, and transmit power, computing resources, and bandwidth allocation are randomly selected. This method serves as a lower bound for performance comparison, illustrating the advantage of learning-based optimization.

5.2. Overall algorithm comparison

In this section, we compare the performance of our proposed CPO-MADDPG algorithm with several baseline methods, including MADDPG, MATD3, MAPPO, LMADDPG, and Random, across three key performance metrics: reward, energy consumption, and safe offload rate.

5.2.1. Reward comparison

To evaluate the overall learning performance, we compare the convergence behavior of CPO-MADDPG with the baseline methods. Fig. 4 plots the average reward against the number of training episodes. The proposed CPO-MADDPG algorithm achieves the fastest convergence rate

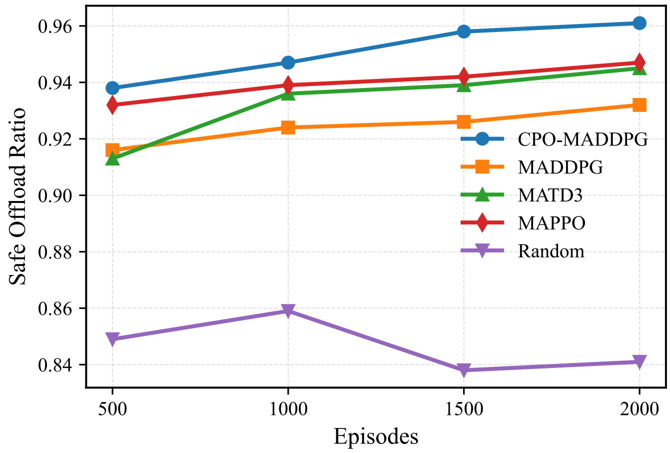

Fig. 6. Safe Offload Rate Comparison.

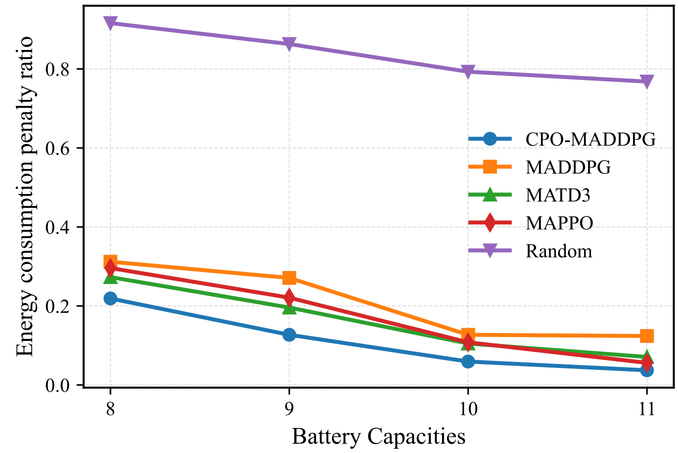

Fig. 8. Energy Consumption Comparison.

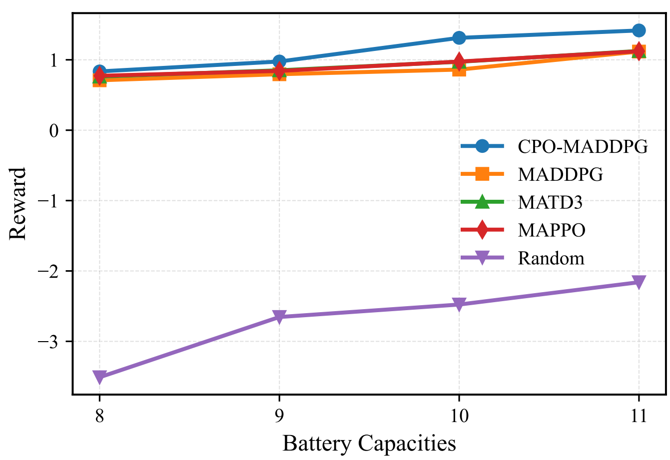

Fig. 7. Reward Comparison.

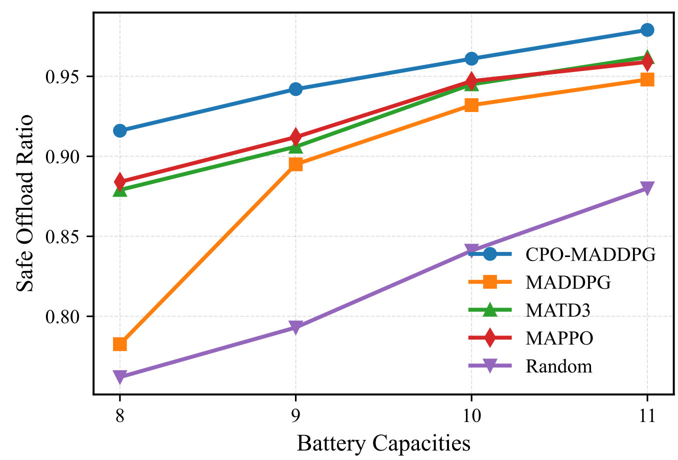

Fig. 9. Safe Offload Rate Comparison.

and the highest average reward among all algorithms. The early training phase shows large fluctuations in MADDPG and MAPPO, which are due to unstable policy updates and lack of explicit constraint modeling. As a constrained reinforcement learning paradigm, LMADDPG yields significantly better long-term reward values compared to unconstrained reinforcement learning methods. In contrast, CPO-MADDPG rapidly stabilizes as the constraint-aware optimization effectively guides the policy search direction. This stability stems from the Lagrangian constrained policy optimization mechanism, which dynamically adjusts the trade-off between reward maximization and constraint satisfaction. The Random policy, as expected, shows high instability and fluctuates near a negative reward value, providing a lower bound for performance comparison.

5.2.2. Energy consumption penalty ratio comparison

Next, we evaluate the energy consumption penalty ratio(ECPR) of different algorithms. Fig. 5 presents the variation of the energy consumption penalty ratio over the training process, which reflects the proportion of energy usage exceeding the predefined constraint limit C_{limit}. During early training, the penalty ratio decreases rapidly and stabilizes below 0.08, demonstrating that the learned policy effectively satisfies the energy constraint while maintaining high task performance. In contrast, traditional MADDPG and MATD3 exhibit higher energy penalty ratios because they lack explicit constraint modeling. While LMADDPG incorporates constraint modeling, its performance still lags behind our method. MATD3 introduces double critics to reduce value estimation bias but treats energy consumption as an implicit component of the reward. MAPPO demonstrates moderate performance but converges more slowly due to its on-policy learning nature. The Random policy, lacking any optimization mechanism, exhibits the highest and most unstable energy consumption penalty, indicating frequent constraint violations.

5.2.3. Safe offload rate comparison

To assess the proposed algorithm's ability to maintain secure communications during task offloading, we introduce the safe offload rate (SOR), which represents the proportion of successfully offloaded video chunks that satisfy the secrecy constraint. Fig. 6 shows the safe offload rate achieved by different algorithms. As expected, traditional DRL algorithms like MADDPG and MATD3 exhibit low safe offload rates since their decision-making policies prioritize throughput and reward maximization, neglecting the security constraint. Consequently, many offloaded tasks violate the secrecy rate requirement, forcing them to be reprocessed locally and degrading overall system performance. In contrast, the proposed CPO-MADDPG algorithm achieves a significantly higher safe offload rate. Overall, CPO-MADDPG achieves superior performance in maintaining secure offloading decisions while optimizing the system's QoS. This feature is especially critical for privacy-sensitive MEC applications, such as real-time video monitoring and intelligent surveillance.

5.2.4. Different number of users comparison

To evaluate the scalability of the proposed algorithm, we investigate the system performance under different numbers of mobile users, $N = \{20, 30, 40, 50\}$. Unless otherwise specified, the total MEC resources, such as bandwidth and edge computing capacity are kept fixed, such

Table 3
Impact of the Number of Users on System Performance.

Number of Users(N)	Reward	ECPR[%]	SOR[%]
20	1.27	0.415	86.28
30	1.22	0.623	86.17
40	1.17	0.931	85.97
50	1.13	1.236	85.90

that increasing N results in a higher level of multi-user contention. We report the converged average reward, ECPR, and SOR. The results are summarized in Table 3.

As number of users increases, the average reward decreases from 1.27 to 1.13, which is mainly due to intensified competition for limited communication and computation resources. Meanwhile, ECPR slightly increases, indicating that a denser user population leads to more aggressive actions and thus more frequent energy-budget pressure. The SOR remains consistently high (above 85%) with only a mild decrease, demonstrating that the proposed secrecy-aware policy can maintain robust secure offloading behavior even in crowded MEC scenarios. Overall, the results confirm the scalability of CPO-MADDPG under increased system load.

5.3. Comparison of different maximum battery capacity and replenishment power

In this section, we will compare the overall performance evaluation of different maximum battery capacity and the overall impact of replenishing different amounts of power at each moment on system performance.

5.3.1. Comparison of different maximum battery capacity

To further investigate the adaptability of the proposed algorithm under varying energy constraints, we evaluate the performance of different algorithms with respect to maximum battery capacity ranging from 8 to 11 units. Furthermore, because the maximum battery capacity varies, the corresponding C_limit also varies, corresponding to 0.4, 0.45, 0.5, and 0.55 respectively. Figs. 7–9 present the comparison results in terms of reward, energy consumption, and safe offload rate.

As shown in Fig. 7, the proposed CPO-MADDPG algorithm consistently achieves the highest average reward across all battery capacities. When the available energy is limited, CPO-MADDPG still maintains a stable reward, while traditional algorithms such as MADDPG and MATD3 experience a sharp degradation due to their lack of adaptive constraint handling.

Fig. 8 shows the corresponding energy consumption penalty ratio. The CPO-MADDPG exhibits the lowest energy consumption under all maximum battery capacity, demonstrating its superior capability to maintain operation within the energy constraint boundary C_{limit}. In contrast, conventional algorithms consume excessive energy when the battery capacity is small, as they lack explicit constraint modeling and rely solely on reward shaping.

Finally, Fig. 9 presents the safe offload rate comparison under different energy levels. The CPO-MADDPG consistently outperforms all baselines, maintaining over 90% safe offloading even when the energy budget is tight. This indicates that the algorithm not only learns energy-efficient task offloading strategies but also ensures data transmission security by avoiding offloading in low-secrecy conditions. MAPPO and MATD3 exhibit noticeable degradation in safe offload rate under low battery conditions, while the random policy performs worst due to the lack of any adaptive strategy. LMADDPG's secure offloading rate is improved compared to other algorithms due to the addition of constraints.

5.3.2. Comparison of different replenishment power

In this section, we evaluate the impact of varying energy replenishment levels on the overall system performance under different conditions. Energy replenishment is uniformly distributed, and at each time

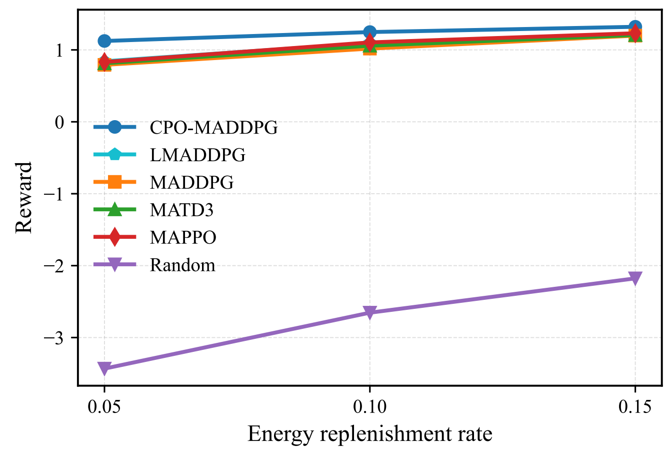

Fig. 10. Comparison of rewards at different energy replenishment rates.

t, energy is replenished at a fixed rate based on the maximum battery capacity. Specifically, we compare the performance of the proposed algorithm with energy replenishment rates of 0.05, 0.10, and 0.15 at each time step,assuming a battery capacity threshold of 8 units. This allows us to analyze how different levels of energy replenishment influence the reward, energy consumption, and safe offloading rate.

The results of the comparison are shown in Figs. 10–12, which illustrate the effects of different energy replenishment levels on reward, energy consumption, and the safe offload rate, respectively.

Fig. 10 shows the comparison of the average reward achieved by the different replenishment rates. As expected, a higher energy replenishment rate leads to a consistently higher reward, as the system can offload tasks more efficiently without hitting energy constraints. The replenishment rate of 0.05, on the other hand, shows a slower convergence with slightly lower rewards due to the limited energy available for offloading tasks. The rate of 0.10 strikes a balance between energy availability and reward maximization, showing better performance than 0.05 but slightly lower than 0.15. This suggests that replenishment plays a crucial role in the task offloading efficiency and overall performance.

Fig. 11 presents the variation of the energy consumption penalty ratio during training, which measures the proportion of energy usage exceeding the predefined constraint C_{limit}. The algorithm with a replenishment rate of 0.15 exhibits the lowest penalty ratio, indicating that it is able to stay within the energy budget more effectively. As the replenishment rate decreases, the penalty ratio increases, especially at the 0.05 replenishment level, which shows frequent energy violations. The replenishment rate of 0.10 demonstrates a moderate penalty, indicating better energy efficiency than 0.05, but still occasionally exceeding the energy constraints. The dynamic adjustment of the Lagrangian multiplier λ during training ensures that the system maintains energy efficiency while optimizing task performance.

Fig. 12 shows the safe offload rate, which represents the proportion of offloaded tasks that meet the secrecy constraint. As the energy replenishment rate increases, the safe offload rate also improves, as higher replenishment allows for more stable offloading decisions. The rate of 0.15 achieves the highest safe offload rate, demonstrating the algorithm's ability to maintain secure offloading decisions even under high energy availability. In contrast, the rate of 0.05 exhibits a lower safe offload rate, as insufficient energy leads to more frequent task reprocessing due to violation of the secrecy constraint.

5.4. Impact of eavesdropping intensity

To evaluate the robustness of the proposed confidentiality constraint offloading mechanism, we investigate different channel gains for the eavesdropper, thus comparing the system's performance under

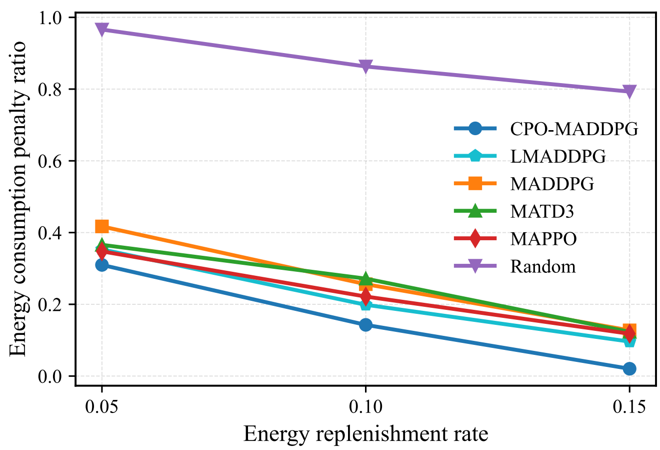

Fig. 11. Comparison of ECPR at different energy replenishment rates.

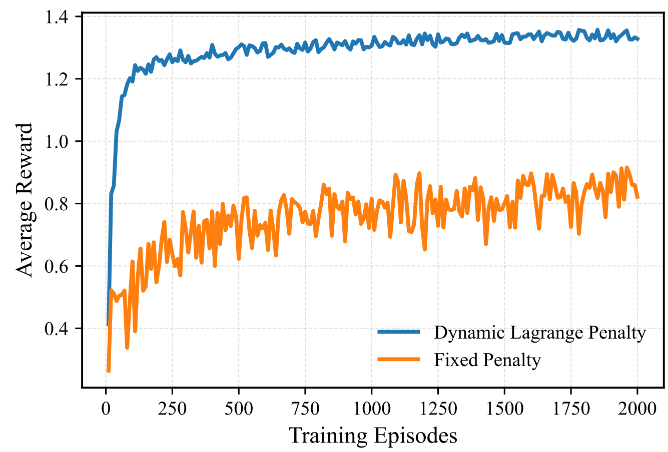

Fig. 13. Comparison of Different Penalty.

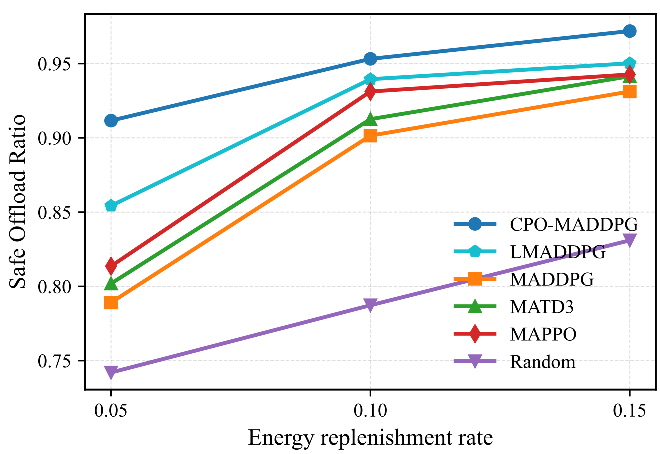

Fig. 12. Comparison of SOR at different energy replenishment rates.

Table 4
Impact of Eavesdropping Strength on Secrecy-Constrained Offloading.

Attacker Strength[$g_{\min}, g_{\max}$]	Reward	ECPR[%]	SOR[%]
Weak[8, 10]	1.27	0.618	86.28
Moderate[6, 12]	1.14	1.369	81.31
Strong[4, 14]	1.03	4.012	74.12

different eavesdropping intensities. We abstractly characterize the attacker's capability using the eavesdropper's channel gain, generating the true gain as $g_{n_k}(t) \sim \mathcal{U}[g_{\min}, g_{\max}]$, while the estimated gain $g^e_{n_k}(t)$ is sampled within the bounded error set $|g_{n_k}(t) - g^e_{n_k}(t)| \le \rho$. By varying the sampling range $[g_{\min}, g_{\max}]$, we simulate three eavesdropping intensities: weak, moderate, and strong. Wider intervals correspond to stronger, more uncertain eavesdropping conditions, leading to stricter confidentiality constraints.

Similar to the previous results, we report three metrics: the average reward for eventual convergence, the ECPR, and SOR. Table 4 summarizes the performance under different eavesing intensities. As the eavesing intensity increases (i.e., the range of $[g_{\min}, g_{\max}]$ becomes wider and more unfavorable), the security offload rate consistently decreases from 86.28% to 74.12%, indicating that the feasible security offload region gradually shrinks in the face of more powerful attackers.Meanwhile, the average reward also decreases from 1.27 to 1.03, reflecting the performance-security tradeoff brought about by stricter confidentiality constraints. Furthermore, regarding the energy consumption penalty rate, stronger eavesing intensity forces more tasks to be executed locally, thus consuming more energy and leading to a gradual increase in the energy consumption penalty rate. These results confirm that the proposed method can effectively adjust its offload decisions according to different eavesing conditions and maintain robust confidentiality-aware behavior.

5.5. Ablation experiment

In this section, we conducted ablation experiments across three dimensions, comparing the overall system performance under fixed penalties versus dynamic Lagrange penalties, different critic networks, and different experience buffer.

5.5.1. Comparison of different punishments

To evaluate the effectiveness of the dynamic penalty method, we conducted ablation experiments comparing fixed penalty and dynamic Lagrange penalty. In these experiments, the maximum battery capacity $B^{\max}$ was set to 10 units, and the constraint C_{limit} was defined as 0.05 times $B^{\max}$. During the training with a fixed penalty, the multiplier λ remained constant at 0.5. For the dynamic penalty case, λ was adjusted dynamically based on the constraint violations to balance the relationship between reward maximization and constraint satisfaction.

In Fig. 13, we compare the reward changes during training for both fixed penalty and dynamic Lagrange penalty. It can be observed that the CPO-MADDPG with dynamic penalty not only converges faster but also achieves more stable rewards compared to the fixed penalty. This suggests that the adaptive nature of the dynamic penalty contributes significantly to the stability and effectiveness of the learning process. To further demonstrate the stability of the method, Fig. 14 shows the trend of energy consumption under dynamic Lagrange penalties. It can be seen that as training progresses, the energy consumed to complete the task continuously decreases until it stabilizes. With dynamic penalty, the energy consumption stabilizes below C_{limit} after approximately 300 training rounds, indicating that the Lagrange multiplier effectively penalizes energy violations and guides the agent toward energy-efficient policies. Furthermore, Fig. 15 illustrates the dynamic updates of the Lagrange multiplier λ. As training progresses, λ increases or decreases to appropriately balance the reward and constraint terms. This figure shows that the dynamic adjustment of λ enables the system to converge to a feasible policy without requiring manual tuning of penalty weights.

These results confirm that the proposed dynamic Lagrange penalty significantly improves both the convergence speed and stability of the system, while maintaining energy efficiency and constraint satisfaction, compared to the fixed penalty strategy.

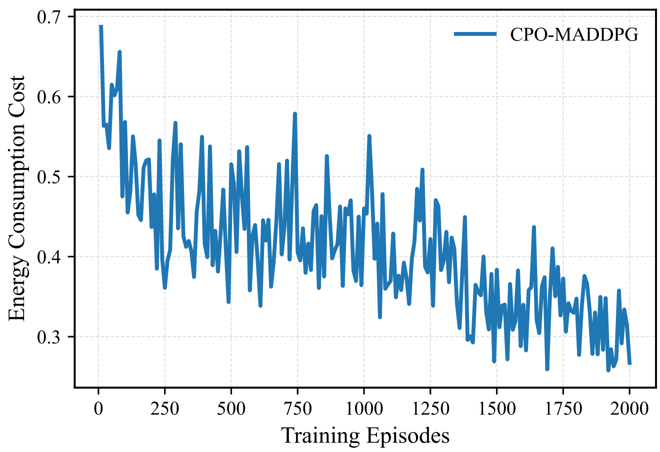

Fig. 14. Energy Cost Changes with Training Episodes.

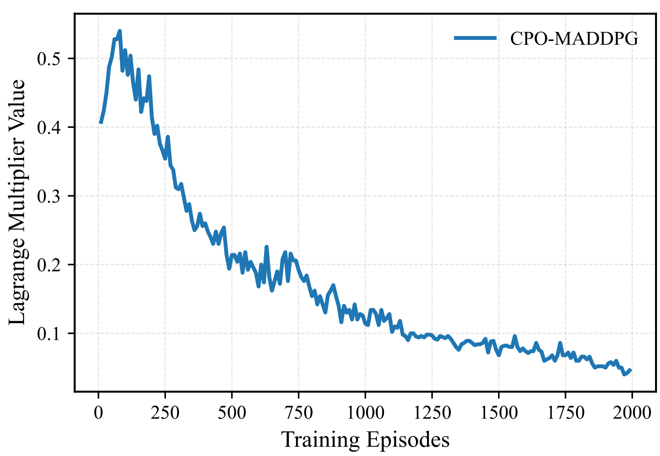

Fig. 15. Lagrange Multiplier Changes.

5.5.2. *Comparison of critic architecture*

To verify the effectiveness of the proposed two-headed critic and to address the potential negative interference between reward and cost estimation, we conduct an ablation study by comparing three critic designs under the same training settings:

- **Ours:** A single critic network with a shared feature extractor and two task-specific heads that output $Q_r(s,a)$ and $Q_c(s,a)$, respectively.
- **Fully Separate Critics:** Two independent critic networks are used: one critic estimates $Q_r(s,a)$ and the other estimates $Q_c(s,a)$, each with its own feature extractor and parameters.
- **Single Critic Multi-Output:** A single critic network is used with a fully shared backbone, and the final The layer directly outputs a 2-dimensional vector $[Q_r(s,a), Q_c(s,a)]$ without any task-specific head branches.

Fig. 16 presents the performance comparison across these three architectures. It is evident that our method significantly outperforms both the fully separate critics and the single critic multi-output design. Specifically, our approach achieves faster convergence and better overall performance, both in terms of reward maximization and efficient constraint handling. These results highlight the advantages of using a shared feature extractor while maintaining task-specific outputs, thus improving training stability and reducing the risk of gradient interference between the two objectives.

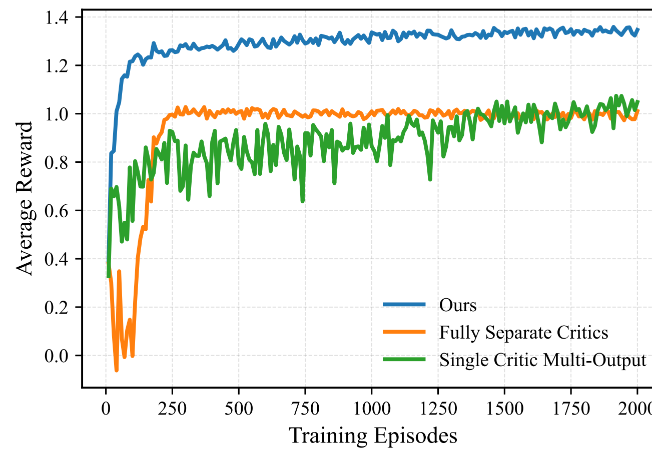

Fig. 16. Comparison of different critics.

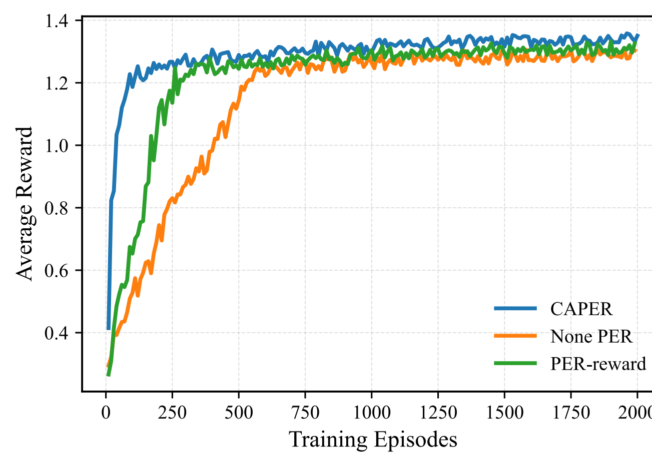

Fig. 17. Comparison of different Buffer.

5.5.3. *Comparison of different PER*

To validate the effectiveness of our CAPER method, we compared it with completely randomized experience replay (None-PER) and methods that only use reward as the priority experience replay condition (PER-reward). Fig. 17 illustrates the impact of these three methods on the overall reward. It can be seen that None-PER, due to the lack of any priority experience replay, converges significantly slower than the other two methods that do use priority experience replay. While our CAPER method doesn't lead by a large margin in final convergence value, it converges significantly faster than PER-reward, compared to PER-reward, which only uses reward value as the priority experience replay condition.

5.6. *Simulation results*

The overall simulation results demonstrate that the proposed CPO-MADDPG algorithm consistently outperforms MADDPG, MATD3, MAPPO, and Random across multiple evaluation metrics. Specifically, it achieves:

- **Faster convergence and higher rewards**, benefiting from two-headed critic learning and constraint-aware gradient optimization.
- **Superior energy efficiency**, due to dynamic λ regulation and cost-aware policy learning.
- **Higher security assurance**, maintained by constraint modeling and PER near the secrecy boundary.
- **Strong adaptability**, remaining robust across various maximum battery capacity and network conditions.

These findings confirm that CPO-MADDPG effectively unifies reward maximization, energy efficiency, and communication security into a cohesive DRL framework, achieving high-quality, secure, and sustainable task offloading for next-generation MEC systems.

6. Conclusion

In this paper, we addressed the problem of secure and energy-efficient video task offloading in MEC environments, where multiple devices compete for limited communication and computational resources under stringent energy and latency constraints. To overcome the limitations of traditional DRL-based methods, which rely solely on reward shaping and struggle to balance energy efficiency, security, and task performance, we proposed a novel CPO-MADDPG algorithm. This algorithm operates within a multi-agent constrained optimization framework.

We extended the traditional Markov game to a constrained Markov game formulation, introducing a unified learning mechanism that integrates constrained policy optimization with deep deterministic policy gradients. Specifically, CPO-MADDPG employs a two-headed critic architecture to estimate both task rewards and energy costs. It also utilizes a dual-channel policy gradient mechanism, which decouples reward maximization from constraint regulation. The dynamically updated Lagrange multiplier guides the policy toward feasible regions, while the CA-PER method enhances learning efficiency and stability, particularly near constraint boundaries.

Our extensive simulation results demonstrate that CPO-MADDPG outperforms baseline algorithms, such as MADDPG, MATD3, and MAPPO. It achieves a 8.5% faster convergence, a 6.1% lower energy consumption penalties, and a 9.6% higher long-term rewards. Additionally, CPO-MADDPG consistently maintains high safe offload rates and stable performance across varying battery capacities, showcasing its ability to adapt to dynamic MEC environments.

In summary, the proposed CPO-MADDPG framework provides a scalable and efficient solution for secure, energy-efficient, and intelligent task offloading. Future work will focus on extending this framework to collaborative multi-edge systems and incorporating model-based predictive optimization and causal reasoning. These advancements will enhance interpretability and generalization, ensuring that the framework is robust and applicable to complex real-world edge computing scenarios.

CRediT authorship contribution statement

Hongyi Chen: Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization; **Yun Peng:** Writing – review & editing, Visualization, Validation, Software, Methodology; **Chen Wei:** Writing – review & editing, Validation, Investigation; **Tao Meng:** Writing – review & editing, Visualization, Software, Resources, Project administration; **Wei Ai:** Writing – review & editing, Visualization, Validation, Supervision, Software, Resources, Project administration; **Zhixiong He:** Writing – review & editing, Visualization, Validation; **Keqin Li:** Writing – review & editing, Validation, Supervision, Methodology.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62372478, 62472165), the Hunan Provincial Natural Science Foundation General Project (Grant No. 2025JJ50380), and the Hunan Provincial Natural Science Foundation Youth Project (Grant No. 2025JJ60420).

References

[1] M.A. Khan, E. Baccour, Z. Chkirbene, A. Erbad, R. Hamila, M. Hamdi, M. Gabbouj, A survey on mobile edge computing for video streaming: opportunities and challenges, IEEE Access 10 (2022) 120514–120550.
[2] M. Mahmoud, S. Rizou, A.S. Panayides, N.V. Kantartzis, G.K. Karagiannidis, P.I. Lazaridis, Z.D. Zaharis, A survey on optimizing mobile delivery of 360 videos: edge caching and multicasting, IEEE Access 11 (2023) 68925–68942.
[3] J. Chen, H. Xing, Z. Xiao, L. Xu, T. Tao, A DRL agent for jointly optimizing computation offloading and resource allocation in MEC, IEEE Int. Things J. 8 (24) (2021) 17508–17524.
[4] H. Ding, Z. Zhao, H. Zhang, W. Liu, D. Yuan, DRL based computation efficiency maximization in MEC-enabled heterogeneous networks, IEEE Trans. Veh. Technol. 73 (10) (2024) 15739–15744.
[5] S. Iftikhar, S.S. Gill, C. Song, M. Xu, M.S. Aslanpour, A.N. Toosi, J. Du, H. Wu, S. Ghosh, D. Chowdhury, et al., AI-based fog and edge computing: a systematic review, taxonomy and future directions, Int. Things 21 (2023) 100674.
[6] H. Li, K.D.R. Assis, S. Yan, D. Simeonidou, DRL-based long-term resource planning for task offloading policies in multiserver edge computing networks, IEEE Trans. Netw. Serv. Manage. 19 (4) (2022) 4151–4164.
[7] G. Ma, X. Wang, M. Hu, W. Ouyang, X. Chen, Y. Li, DRL-based computation offloading with queue stability for vehicular-cloud-assisted mobile edge computing systems, IEEE Trans. Intell. Veh. 8 (4) (2022) 2797–2809.
[8] M. Sharma, A. Tomar, A. Hazra, Edge computing for industry 5.0: fundamental, applications, and research challenges, IEEE Int. Things J. 11 (11) (2024) 19070–19093.
[9] Y. Liu, L. Jiang, Q. Qi, S. Xie, Energy-efficient space–air–ground integrated edge computing for internet of remote things: a federated DRL approach, IEEE Int. Things J. 10 (6) (2022) 4845–4856.
[10] J. Li, B. Gu, Z. Qin, Y. Han, Graph tasks offloading and resource allocation in multi-access edge computing: a DRL-and-optimization-aided approach, IEEE Trans. Network Sci. Eng. 10 (6) (2023) 3707–3718.
[11] J. Zhang, B. Chen, Y. Zhao, X. Cheng, F. Hu, Data security and privacy-preserving in edge computing paradigm: survey and open issues, IEEE Access 6 (2018) 18209–18237.
[12] S. Wang, J. Li, G. Wu, H. Chen, S. Sun, Joint optimization of task offloading and resource allocation based on differential privacy in vehicular edge computing, IEEE Trans. Comput. Social Syst. 9 (1) (2021) 109–119.
[13] X. Xu, Q. Wu, L. Qi, W. Dou, S.-B. Tsai, M.Z.A. Bhuiyan, Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles, IEEE Trans. Intell. Transp. Syst. 22 (3) (2020) 1787–1796.
[14] H. Hu, W. Song, Q. Wang, R.Q. Hu, H. Zhu, Energy efficiency and delay tradeoff in an MEC-enabled mobile IoT network, IEEE Int. Things J. 9 (17) (2022) 15942–15956.
[15] T. Zhao, F. Li, L. He, Secure video offloading in MEC-enabled IIoT networks: a multicell federated deep reinforcement learning approach, IEEE Trans. Ind. Inf. 20 (2) (2023) 1618–1629.
[16] T. Zhao, L. He, X. Huang, F. Li, DRL-based secure video offloading in MEC-enabled IoT networks, IEEE Int. Things J. 9 (19) (2022) 18710–18724.
[17] P. Ranaweera, A.D. Jurcut, M. Liyanage, Survey on multi-access edge computing security and privacy, IEEE Commun. Surv. Tutor. 23 (2) (2021) 1078–1124.
[18] Y. Xu, M.Z.A. Bhuiyan, T. Wang, X. Zhou, A.K. Singh, C-fdrl: context-aware privacy-preserving offloading through federated deep reinforcement learning in cloud-enabled IoT, IEEE Trans. Ind. Inf. 19 (2) (2022) 1155–1164.
[19] P. Zhang, P. Gan, L. Chang, W. Wen, M. Selvi, G. Kibalya, DPRL: Task offloading strategy based on differential privacy and reinforcement learning in edge computing, IEEE Access 10 (2022) 54002–54011.
[20] J. Li, S. Zhang, J. Geng, J. Liu, Z. Wu, H. Zhu, A differential privacy based task offloading algorithm for vehicular edge computing, IEEE Int. Things J. (2025).
[21] W. Zheng, Z. Li, A distributed task offloading optimization model and algorithm with privacy protection, in: International Conference on Intelligent Computing, Springer, 2025, pp. 480–496.
[22] H. Wang, X. Li, Z. Shen, P. Liu, Collaborative privacy-Preserving based on LP and orp-Drl mechanism for task offloading in mec, Available at SSRN 5153819.
[23] B. Jiang, Q. He, P. Liu, S. Maharjan, Y. Zhang, Blockchain empowered secure video sharing with access control for vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 24 (9) (2023) 9041–9054.
[24] D.K. Sajnani, L. Xiaoping, A.R. Mahesar, K.Y. Rajput, W. Li, Deep reinforcement learning based security-Aware computational offloading and resource allocation for MEC systems, in: 2025 28th International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, 2025, pp. 576–581.
[25] Y. Fei, H. Fang, Z. Yan, L. Qi, M. Bilal, Y. Li, X. Xu, X. Zhou, Privacy-Aware edge computation offloading with federated learning in healthcare consumer electronics system, IEEE Trans. Consum. Electron. (2025).
[26] S. Zhang, H. Gu, K. Chi, L. Huang, K. Yu, S. Mumtaz, DRL-based partial offloading for maximizing sum computation rate of wireless powered mobile edge computing network, IEEE Trans. Wireless Commun. 21 (12) (2022) 10934–10948.

[27] Y. Liu, C. Yang, L. Jiang, S. Xie, Y. Zhang, Intelligent edge computing for IoT-based energy management in smart cities, IEEE Netw. 33 (2) (2019) 111–117.
[28] X. Fu, L. Pan, S. Liu, A DRL-Based real-Time video processing framework in cloud-Edge systems, IEEE Int. Things J. (2024).
[29] X. Xu, Y. Song, A deep reinforcement learning-based optimal computation offloading scheme for VR video transmission in mobile edge networks, IEEE Access 11 (2023) 122772–122781.
[30] S. Bi, H. Chen, X. Li, S. Wang, Y. Wu, L. Qian, A two-stage deep reinforcement learning framework for mec-enabled adaptive 360-degree video streaming, IEEE Trans. Mob. Comput. (2024).
[31] S. Wang, J. Yang, S. Bi, Adaptive video streaming in multi-tier computing networks: joint edge transcoding and client enhancement, IEEE Trans. Mob. Comput. 23 (4) (2023) 2657–2670.
[32] X. Mi, H. He, H. Shen, A multi-agent rl algorithm for dynamic task offloading in d2d-mec network with energy harvesting, Sensors 24 (9) (2024) 2779.
[33] A. Lotfolahi, H.-W. Ferng, DRL-Based Resource allocation in NOMA-Aided industrial IoT towards energy productivity maximization, IEEE Trans. Network Sci. Eng. (2025).
[34] W. Zhou, Y. Lu, W. Pan, Z. Chen, J. Qin, Dynamic bitrate adaptation and bandwidth allocation for MEC-Enabled video streaming, IEEE Commun. Lett. (2024).
[35] X. Zhou, J. Zeng, S. Ge, X. Liu, T. Qiu, Collaborative video streaming with super-Resolution in multi-User MEC networks, IEEE Trans. Mob. Comput. (2024).
[36] P. Zhao, Z. Yang, G. Zhang, Personalized and differential privacy-aware video stream offloading in mobile edge computing, IEEE Trans. Cloud Comput. 12 (1) (2024) 347–358.
[37] T.Z. Gebrekidan, S. Stein, T.J. Norman, Combinatorial client-master multiagent deep reinforcement learning for task offloading in mobile edge computing, arXiv preprint arXiv:2402.11653 (2024).