


Article

Minimizing Task Age upon Decision for Low-Latency MEC Networks Task Offloading with Action-Masked Deep Reinforcement Learning

Zhouxi Jiang, Jianfeng Yang *  and Xun Gao 

Electronic Information School, Wuhan University, Wuhan 430072, China; jiangzhouxi@whu.edu.cn (Z.J.); gaoxun@whu.edu.cn (X.G.)

* Correspondence: yjf@whu.edu.cn

Abstract: In this paper, we consider a low-latency Mobile Edge Computing (MEC) network where multiple User Equipment (UE) wirelessly reports to a decision-making edge server. At the same time, the transmissions are operated with Finite Blocklength (FBL) codes to achieve low-latency transmission. We introduce the task of Age upon Decision (AuD) aimed at the timeliness of tasks used for decision-making, which highlights the timeliness of the information at decision-making moments. For the case in which dynamic task generation and random fading channels are considered, we provide a task AuD minimization design by jointly selecting UE and allocating blocklength. In particular, to solve the task AuD minimization problem, we transform the optimization problem to a Markov Decision Process problem and propose an Error Probability-Controlled Action-Masked Proximal Policy Optimization (EMPPO) algorithm. Via simulation, we show that the proposed design achieves a lower AuD than baseline methods across various network conditions, especially in scenarios with significant channel Signal-to-Noise Ratio (SNR) differences and low average SNR, which shows the robustness of EMPPO and its potential for real-time applications.



Citation: Jiang, Z.; Yang, J.; Gao, X. Minimizing Task Age upon Decision for Low-Latency MEC Networks Task Offloading with Action-Masked Deep Reinforcement Learning. *Sensors* **2024**, *24*, 2812. <https://doi.org/10.3390/s24092812>

Academic Editors: Ricardo S. Alonso Rincón, Sara Rodríguez and Iñaki Fernández Pérez

Received: 14 March 2024

Revised: 18 April 2024

Accepted: 26 April 2024

Published: 28 April 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: low-latency mobile edge computing; age upon decision; finite blocklength regime; deep reinforcement learning; maskable proximal policy optimization

1. Introduction

Mobile Edge Computing (MEC) has been recognized as a key enabling technology supporting ultra-reliable and latency-sensitive applications in the future Sixth-Generation Mobile Networks (6G) [1,2], such as autonomous driving [3], simultaneous localization and mapping [4], intelligent manufacturing [5], and unmanned aerial vehicle [6]. For instance, in autonomous driving, vehicles need to promptly transmit and process real-time data from various sources, including sensors on vehicles and other roadside servers, to achieve comprehensive awareness of the road environment and make reliable decisions for the following actions. Since vehicles may travel at high speeds, the process needs to be completed within tens of milliseconds and with high reliability. Considering the limited performance and resources of User Equipment (UE), and dynamic wireless channels, Ultra-Reliable Low-Latency Communications (URLLC) in MEC networks is challenging.

In the MEC network, an edge server typically serves multiple UEs. Given the finite communication and computation resources, there is often competition among UEs. Therefore, offloading decisions and resource allocation impact system performance and should be taken into account during system design [7–12]. In [7,8], the authors optimize latency in the MEC network by controlling the offloading strategy and power allocation. In addition to making an offloading decision, the authors in [9] allocate the CPU frequency of the MEC server and assign wireless bandwidth for the transmission. The work in [10] provides an optimal design to minimize the overall error probability by server selection and time allocation. The authors of [11] introduce a fast futures-enabled resource trading mechanism

to determine optimal transmission power and further propose a hybrid market approach that integrates futures and spot trading to facilitate resource trading in [12].

However, all of these studies are based on the assumption of infinite blocklength, which means the transmission of data packets is considered arbitrarily reliable at Shannon's capacity. To meet low latency requirements, data packets wirelessly uploading/offloading in MEC networks are more likely to operate via short block codes, so-called communications in the Finite Blocklength (FBL) regime [13]. In the FBL regime, transmission error possibly occurs even when the coding rate is set to be lower than the Shannon capacity. When transmitting data packets with shorter blocklength to reduce transmission delay, the probability of transmission error increases correspondingly. Therefore, minimizing latency with a reliability guarantee under the FBL regime should be carefully investigated.

In the past few years, a set of works have taken the FBL impacts into consideration in the performance analysis and system design for MEC networks [14–17]. The study in [14] introduces an energy-efficient algorithm for dynamic computation offloading in a multi-access edge computing scenario, focusing on delay-critical applications by incorporating URLLC through FBL and reliability constraints to manage radio and computational resources jointly. The authors of [15] propose an efficient algorithm for an unmanned aerial vehicle-enabled MEC system with URLLC-based offloading to minimize the maximum computation latency, considering the FBL transmission and its impact on the data rate. Moreover, the work in [16] proposes an MEC-aided integrated sensing and communication scheme that leverages short-packet transmissions to efficiently offload radar-sensing data to an edge-server, ensuring low latency and high reliability for radar data analysis while minimizing system energy consumption. In [17], the authors investigate the effects of short-packet transmission on the radio resource allocation and minimize the energy cost for mission-critical Internet-of-Things (IoT) in an MEC system. Some authors have also combined FBL with various communication technologies, such as non-orthogonal multiple access and retransmission, to analyze the impact on system performance [18–21]. The authors in [18] provide a design that maximizes the effective capacity via FBL transmission. In [19], a joint optimization problem is formulated by determining the user grouping and allocating blocklength. In [20], the authors propose energy-efficient retransmission schemes for MEC networks with a hybrid automatic repeat request, optimizing the number of retries and transmission parameters under FBL constraints to minimize energy consumption for latency-critical tasks. In [21], the authors allocate blocklength to improve the average end-to-end reliability through a deep reinforcement learning approach. However, existing studies often focus on the transmission delay under the FBL regime. They tend to overlook the queuing delay, especially when tasks involve multiple related data packets.

Since transmission delay does not fully capture the total time taken for transmitting packets, especially ignoring the waiting time of packets at the source, some researchers have introduced the Age of Information (AoI) [22]. The AoI concept, which includes the waiting time of packets at the source, offers a more accurate assessment of packet freshness.

Recently, a set of studies have analyzed the AoI performance of networks operating with FBL codes. The authors in [23,24] present a novel study for downlink cellular networks and orthogonal frequency division multiple access systems to minimize AoI, considering the impact of FBL. In [25], authors optimize the tail distribution of AoI in vehicular networks by employing extreme value theory. The work in [26] investigates the correlation and joint optimization of AoI and peak AoI in a last-come first-served system with retransmission and non-preemption policies. Some other recent studies have analyzed the AoI within the MEC networks. In [27], the authors investigate the average AoI in a wireless-powered MEC system and derive a closed-form expression for it. In [28], the problem of minimizing AoI in an MEC system for computation-intensive status updates is addressed with the no-wait policy and interval-wait policy. The authors of [29] present an AoI-based optimization strategy for computation offloading and transmission scheduling in MEC-enabled IoT networks. In [30], the authors analyze the average AoI and the average peak AoI in a multi-user MEC system, where a base station transmits computation-intensive packets

to UEs. However, in some scenarios, data packets are used for decision-making after transmission. The freshness of the data at the time of decision is more critical than at the moment transmission is completed. The AoI does not capture the packets' freshness for decision-making purposes.

More recently, the concept of Age upon Decision (AuD) has been introduced in [31]. The AuD measures how old the information is when it is used to make a decision, whereas AoI measures the information received. The authors investigate the impact of scheduling update arrival and decision-making processes on the average AuD in [32,33]. Then, the authors of [34] explored the AuD in IoT-based wireless networks with truncated hybrid automatic repeat requests, focusing on the impact of FBL and multiple sources. Although much of the research has looked at the AuD for data packets, the task AuD that involves multiple data packets still requires further investigation. To the best of our knowledge, there is also a missing analysis and optimization of a task's AuD with multiple related data packets in low-latency MEC networks under the FBL regime.

In this paper, we consider task transmission and computation in a low-latency MEC network with multiple UEs and one MEC edge server. To minimize the average task AuD, we allocate finite blocklength resources to UEs and decide the transmission moment. We propose an action-masked Proximal Policy Optimization (PPO) algorithm that adapts to time-varying independent channels and randomly arriving tasks while ensuring reliable task transmission. Our main contributions are summarized as follows:

- We introduce a low-latency MEC network where tasks are composed of multiple data packets transmitted by various UEs through wireless channels to an edge server under the FBL regime. We analyze the impact of blocklength allocation and UE selection on the error probability of data packet transmission and the task AuD.
- We formulate an optimization problem to minimize the average task AuD by jointly allocating blocklength and selecting UEs while ensuring error probability constraints. Subsequently, we transform this into a UE selection problem that adheres to the error probability constraints dictated by the FBL regime.
- We propose an Error Probability-Controlled Action-Masked Proximal Policy Optimization (EMPPPO) algorithm considering dynamic task generation and random fading channels. By masking actions that cannot satisfy error probability constraints, our algorithm prevents the exploration of ineffective actions, thereby enhancing efficiency.
- Simulation results are provided to validate the performance of our proposed method compared to server baseline methods, especially the standard DRL methods. We demonstrate the robustness of our method under various network conditions, such as varying task arrival probabilities, channel correlation coefficients, and Signal-to-Noise Ratio (SNR).

The rest of the paper is organized as follows: In Section 2, we describe the low-latency MEC system model and FBL performance and task AuD. We state the optimization problem of minimizing the task AuD and present an action-masked DRL algorithm in Section 3. Then, we evaluate the performance and show the simulation results in Section 4. Finally, we conclude our work in Section 5.

2. Preliminaries

In this section, we introduce the system model of the low-latency MEC network and FBL regime in detail. Then, we introduce and analyze the task AuD in a two UEs low-latency MEC scenario.

2.1. System Model

As shown in Figure 1, the system model of the low-latency MEC network we consider is composed of a set of UEs $\mathcal{U} = \{1, 2, \dots, K\}$ and one MEC server. We discretize time into frames of consistent length. At the beginning of each frame, the task arrives randomly. A task consists of multiple data packets of different sizes. Each packet is stored in a user's queue and then transmitted to the edge server at a later frame. One frame may transmit

one or more packets. Once all data packets of a task have been successfully transmitted, the task can start computation and then be used to decide when the computation is finished.

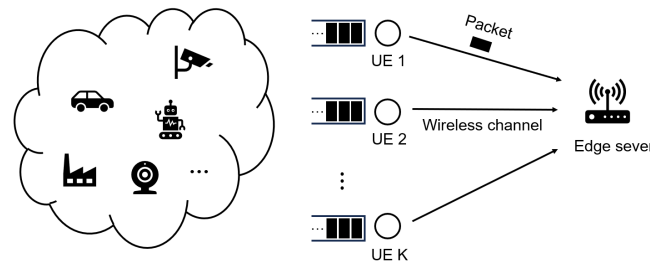


Figure 1. System model of low-latency MEC network.

Specifically, we discretize time into a set of time frames $\mathcal{T} = \{1, 2, \dots, T\}$. Each frame is comprised of n symbols, which are the smallest units required to represent the transmitted data in our communication system. A symbol is not merely a bit, instead, it encapsulates a discrete value or a set of bits according to the modulation scheme used. The duration of each symbol is T_s , which is how long it takes to transmit one symbol. Therefore, the total duration of the frame T_f can be represented as

$$T_f = nT_s \quad (1)$$

Each frame can be split into a communication phase and a computation phase. The frame structure is shown in Figure 2. During the transmission phase, data packets from different UEs are transmitted to the edge server via the wireless channels using time-division multiple access. During the computation phase, we consider joint-packet task scenarios. That is, a task is made up of data packets from multiple UEs. A task's computation and decision-making process can only be initiated when all data packets involved have been transmitted (transmission can be distributed across different frames). Considering that the ability to start decision-making for a task depends mainly on whether the last data packet of the task has been successfully transmitted, we utilize a fixed time length for the computation phase. Tasks in which all packets have been fully transmitted can proceed with calculation and decision-making in the computation phase.

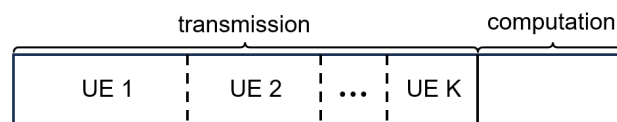


Figure 2. Frame structure.

Data packets may arrive at the beginning of each frame, that is, before the transmission starts. A data packet with packet size $m_{k,t}$ (in bits) arrives at the UE k in frame t with a probability of p , suggesting that the data packet arrivals conform to an On/Off Markov arrival model:

$$P = \begin{cases} p & \text{packet arrives} \\ 1 - p & \text{packet does not arrive} \end{cases} \quad (2)$$

Furthermore, for each UE, a data packet is then cached in an independent First-In-First-Out (FIFO) queue.

Notably, we consider joint task decisions that require multiple data packets. This suggests that for a specific task, its data packets always arrive at various UEs at the same time (e.g., in an autonomous driving scenario, we can realize a comprehensive perception of the road environment at a specific moment by aggregating the data captured from different sensors at the exact moment).

Additionally, due to potential variations in the types of UEs, the sizes of data packets arriving at each UE may differ. We assume that the packet size $m_{k,t}$ is uniformly distributed across the range $[a, b]$, ensuring that each packet size within this interval is equally likely to occur. The probability density function of the packet size $m_{k,t}$ is given by:

$$f(m_{k,t}) = \begin{cases} \frac{1}{b-a} & a \leq m_{k,t} \leq b \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Channels are considered to experience quasi-static Rayleigh fading. Therefore, as the duration of a frame T_f is shorter than the channel coherent time, the channel fading remains constant within the frame but may vary between the current frame and the next. Channels between different UE and edge servers are considered to be independent.

Thus, for channel state $h_{k,t}$, we adopt the widely used Jakes model to depict the correlation of channel states between frames:

$$h_{k,t} = \rho h_{k,t-1} + \sqrt{1 - \rho^2} \Delta h_{k,t} \quad (4)$$

where $h_{k,t}$ is the channel state of frame t between user k and the edge server, $h_{k,t-1}$ is the channel state of frame $t - 1$ between user k and the edge server, $\rho \in [0, 1]$ is the channel correlation coefficient, and $\Delta h_{k,t} \sim \mathcal{CN}(0, 1)$ is a complex Gaussian random variable. The channel state $h_{k,t}$ is modeled as influenced by the previous state $h_{k,t-1}$ through the correlation coefficient ρ and a random component $\sqrt{1 - \rho^2} \Delta h_{k,t}$, which represents the time correlation and randomness of the wireless channel. Hence, we have the received signal-to-noise (SNR) given by

$$\gamma_{k,t} = \bar{\gamma}_k h_{k,t}^2 \quad (5)$$

where $\bar{\gamma}_k$ is the average SNR of channel k . We consider instantaneous perfect Channel State Information (CSI) is available.

2.2. FBL Communication Performance

To achieve low-latency transmission, the data packet is transmitted within the FBL regime, which means the FBL regime should be considered during the transmission phase.

Following the FBL regime [13], the coding rate $r_{k,t}$ for an additive white Gaussian noise channel between UE k and edge server in frame t is shown to have the following approximation:

$$r_{k,t} = \mathcal{R}(\gamma_{k,t}, \varepsilon_{k,t}, n_{k,t}) \approx \mathcal{C}(\gamma_{k,t}) - \sqrt{\frac{V(\gamma_{k,t})}{n_{k,t}}} Q^{-1}(\varepsilon_{k,t}) \quad (6)$$

where $\gamma_{k,t}$ is the instantaneous SNR, $\varepsilon_{k,t}$ is the error probability of transmission, $n_{k,t}$ is the blocklength, $\mathcal{C}(\gamma_{k,t}) = \log_2(1 + \gamma_{k,t})$ is the Shannon capacity, $V(\gamma_{k,t}) = 1 - \frac{1}{(1 + \gamma_{k,t})^2}$ is the channel dispersion, $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt$ is the Gaussian Q-function, and $Q^{-1}(x)$ is its inverse function. From (6), the error probability of the transmission $\varepsilon_{k,t}$ can be expressed as:

$$\varepsilon_{k,t} \approx Q\left(\frac{\mathcal{C}(\gamma_{k,t}) - r_{k,t}}{\sqrt{\frac{V(\gamma_{k,t})}{n_{k,t}}}}\right) \quad (7)$$

For a data packet with packet size $m_{k,t}$ transmitted at the UE k in frame t , the coding rate $r_{k,t}$ can also be expressed as:

$$r_{k,t} = \frac{m_{k,t}}{n_{k,t}} \quad (8)$$

From the analysis above, we know that allocating more blocklength for the transmission of a data packet will reduce the transmission error probability. However, it will also result in longer transmission times. Additionally, the error probability is affected by the time-varying CSI. Therefore, choosing the appropriate blocklength and transmission timing for data packets is important to achieve a trade-off between error probability and transmission time.

2.3. Task AuD

To better understand the task AuD, we use a simple example comprising two UEs and an edge server for low-latency MEC network. Figure 3 shows the task AuD of the network.

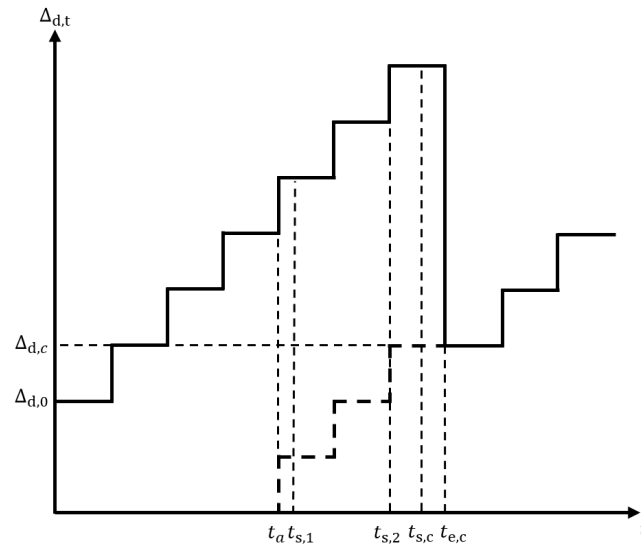


Figure 3. Task AuD of the low-latency MEC network.

As shown in Figure 3, the system's initial AuD is $\Delta_{d,0}$. As time is divided into frames, if no task completion decision is made, the AuD for the next frame will be

$$\Delta_{d,1} = \Delta_{d,0} + T_f \quad (9)$$

Then, a task arrives at the UEs at t_a . This task consists of two packets m_1 and m_2 , which are individually cached in the queues of UE1 and UE2. After waiting in the queue, the packets at both UE1 and UE2 start transmitting to the edge server (may transmit in different frames). Specifically, the packet at UE1 starts transmission at $t_{s,1}$ and the packet at UE2 begins transmission at $t_{s,2}$. According to the system model, packets must be completed within the frame when starting transmission. After the two packets have completed their transmission (i.e., the task transmission is completed), the task begins to compute on the edge server at a subsequent moment, represented as $t_{s,c}$, and the computation ends at $t_{e,c}$. As shown in Figure 2, the computation phase of the task consumes a fixed time length. The computation finishes at the end of the frame with making the task decision. Since the task is decided, we have the new task AuD of the network in the current frame:

$$\Delta_{d,c} = t_{e,c} - t_a \quad (10)$$

Equation (10) shows that the new task AuD is determined by the time that all data packets of the task complete transmission and the task completes computation. Thus, for the task AuD $\Delta_{d,t}$ in frame t , the task AuD $\Delta_{d,t+1}$ in the next frame is determined by:

$$\Delta_{d,t+1} = \begin{cases} \Delta_{d,t} + T_f & \text{if no task decided in frame } t \\ t_{e,c} - t_a & \text{if the task decided in frame } t \end{cases} \quad (11)$$

where T_f is the frame duration, $t_{e,c}$ is the timestamp when the task computation is completed and the decision is made, t_a is the timestamp when the task arrives at the UEs.

Therefore, for an MEC network that has gone through N frames, its average task AuD is defined as:

$$\bar{\Delta}_d = \frac{1}{N} \sum_{t=1}^N \Delta_{d,t} \quad (12)$$

From the example, it can be seen that, compared to the AuD of a single data packet, which only depends on when the packet is used for decision-making after being transmitted to the edge server, Task AuD requires that all packets of a task reach the edge server before they can participate in decision-making together. The Task AuD of a task is largely influenced by the packet that arrives last at the edge server.

3. DRL-Based Joint User Selection and Blocklength Allocation Design

In this section, we formulate the optimization problem of UE selection and blocklength allocation. Then, we formulate the task offloading problem as a Markov Decision Process (MDP) that can be solved by deep reinforcement learning methods. Finally, we investigate the EMPPPO algorithm in detail.

3.1. Problem Formulation

For a network composed of K UEs and one MEC server, we will decide which packets will be transmitted, and the blocklength to be used for each packet in each frame to minimize long-term average task AuD. Considering that the data packets on each UE are cached in a FIFO queue, we assume each UE can transmit at most one packet within a frame. Therefore, we can use a set to represent the selection of UEs and blocklength allocation in frame t :

$$\mathcal{N}_t = \{n_{1,t}, n_{2,t}, \dots, n_{k,t}\}, n_{k,t} \in \mathbb{N} \quad (13)$$

where $n_{k,t}$ is the blocklength assigned to UE k in frame t . $n_{k,t} = 0$ means that the UE does not transmit packets in this frame. From (13), the optimization problem of task offloading can be formulated as follows:

$$\min_{\mathcal{N}_t, t \in \mathcal{T}} \bar{\Delta}_d \quad (14a)$$

$$\text{s.t.} \quad \sum_{k=1}^K n_{k,t} = n_c, \forall i \in \mathcal{T} \quad n_{k,t} \in \mathbb{N} \quad (14b)$$

$$\varepsilon_{k,t} \leq \varepsilon_{max}, \forall k \in \mathcal{U}, t \in \mathcal{T} \quad (14c)$$

$\bar{\Delta}_d$ is the optimization objective, defined as minimizing the average AuD in Equation (12). The constraint (14b) guarantees that the total blocklength equals the available blocklength for the transmission phase in a frame. The constraint (14c) ensures that the error probability for each transmitted packet does not exceed the error probability constraint ε_{max} .

From (7) and (8), it is evident that the error probability monotonically decreases as the blocklength increases. Thus, for a given error probability constraint ε_{max} , with the known SNR $\gamma_{k,t}$ and data packet size $m_{k,t}$, we can determine the optimal blocklength $n_{o,k,t}$ for transmission such that the packets are transmitted exactly meeting the error probability constraint.

Therefore, when we select a UE, the transmitted blocklength is determined uniquely. By using the binary decision variables, we can transform the joint problem of UE selection and blocklength allocation into a UE selection problem. At frame t , the set of binary decision variables used to determine whether the UE transmits is:

$$\mathcal{X}_t = \{x_{1,t}, x_{2,t}, \dots, x_{K,t}\}, x_{K,t} \in \{0, 1\} \quad (15)$$

Then, we can transform the optimization problem into:

$$\min_{\mathcal{X}_{t,t} \in \mathcal{T}} \overline{\Delta_d} \quad (16a)$$

$$\text{s.t.} \quad \sum_{k=1}^K n_{o,k,t} \leq n_c, \forall t \in \mathcal{T} \quad (16b)$$

$$\mathcal{X}_t = \{x_{1,t}, x_{2,t}, \dots, x_{K,t}\}, x_{K,t} \in \{0, 1\} \quad (16c)$$

Constraint (16b) is the total blocklength constraint for a frame. Constraint (16c) is the set of binary decision variables, where $x_{K,t}$ represents the binary decision for each UE K at frame t , taking a value of either 0 or 1. $x_{K,t} = 1$ means UE K transmits its data packet using the optimal blocklength that precisely meets the error probability constraint in frame t , whereas $x_{K,t} = 0$ means that the UE K will not transmit a data packet in that frame. Given that the selected packets for transmission are transmitted in such a way to exactly meet the error rate constraint, the total blocklength may not exactly equal the blocklength available for transmission in one frame. In real-world scenarios, we allocate the remaining blocklength to the packets to achieve lower transmission error probability.

3.2. MDP Formulation

UE selection problem (16a) can be considered as a sequential decision-making process. At any given frame, we can only access the present and historical states, and make decisions under the current state which will influence future states and decisions. Therefore, obtaining the optimal global solution for this problem is impossible. However, it can be modeled as a MDP and solved by reinforcement learning methods.

Reinforcement Learning tasks are usually modeled by MDP. Each state of the Reinforcement Learning system must have the Markov property, meaning once the current system state is known, future states are solely dependent on it and not on past states. An MDP in Reinforcement Learning can be described with the tuple (S, A, P, R, γ) . S represents the state space, which is the set of all possible states of the environment. A is the action space representing the set of all possible actions the agent can take. P is the set of state transition probabilities when the agent performs an action in a given state. R is the reward function. γ is the discount factor that dictates the degree to which future rewards brought by actions are discounted when calculating cumulative rewards.

The classical reinforcement learning process includes the following steps:

- (1) At every time step t , the agent observes the current environment state $s_t \in S$.
- (2) The agent selects an action a_t according to the policy $\pi(s_t)$ based on the current environmental state.
- (3) The environment transitions to state s_{t+1} due to action a_t and gives a reward r_t to the agent.

The learning process of reinforcement learning is an interactive process between the agent and the environment. It allows the agent to explore in an unfamiliar environment where different actions lead to different rewards. As the agent interacts continuously with the environment, it constantly improves its strategy based on changes in rewards and environmental states. After many trials and errors, it obtains the strategy with the most cumulative rewards.

The agent continuously updates and iterates its strategy according to the rewards obtained from the actions taken in different environmental states in the ongoing interactive process, aiming to obtain as much cumulative reward as possible.

In our low-latency MEC scenario, the definitions of state space, action space, and reward function are as follows.

- **State Space:** The state space is composed of three components, which are the age and size of the oldest data packet in each UE's queue (the oldest data packet will be at

the head of the queue), and the real-time SNR of the channels. Therefore, the state in frame i can be represented as:

$$s_t = \{\Delta_{1,t}, \Delta_{2,t}, \dots, \Delta_{K,t}, m_{1,t}, m_{2,t}, \dots, m_{K,t}, \gamma_{1,t}, \gamma_{2,t}, \dots, \gamma_{K,t}\} \quad (17)$$

where $[\Delta_{1,t}, \Delta_{2,t}, \dots, \Delta_{K,t}]$ is the vector of length K containing the age of the oldest data packet in each UE, $[m_{1,t}, m_{2,t}, \dots, m_{K,t}]$ is the vector of length K containing the size of the oldest data packet in each UE, and $[\gamma_{1,t}, \gamma_{2,t}, \dots, \gamma_{K,t}]$ is the vector of length K containing the real-time SNR of the channels.

- Action Space: From (15), we use a vector of binary decision variables to determine whether the K -th UE transmits in frame t :

$$a_t = \{x_{1,t}, x_{2,t}, \dots, x_{K,t}\}, x_{K,t} \in \{0, 1\} \quad (18)$$

where $x_{K,t} = 1$ means UE K transmits a data packet to an edge server in frame t , and $x_{K,t} = 0$ means UE K does not transmit a data packet in frame t . Since each x can be either 0 or 1, there are a total of 2^n possible actions for a_t .

- Reward Function: To minimize the average task AuD, the reward function we have designed is as follows:

$$r_t = -\alpha \Delta_{d,t} + C \quad (19)$$

where $\Delta_{d,t}$ is the task AuD at frame t , α is a scaling factor that adjusts the sensitivity to changes in AuD, ensuring that the reward is neither too large nor too small, and C is a constant term that provides a fixed uplift to the reward, potentially aiding in learning stability and convergence of the learning process.

3.3. EMPPO Algorithm

EMPPPO comes from the standard PPO algorithm and its adapted version maskable PPO. The PPO algorithm was proposed by John Schulman in 2017 [35] and has been widely adopted due to its simplicity, stability and efficiency. It is a policy gradient method, which provides an improvement to trust region policy optimization [36]. Then, considering scenarios where actions cannot always be selected, the maskable PPO has been proposed in 2022 [37]. By masking nonviable actions, it reduces ineffective exploration by agents, thereby enhancing the efficiency and performance of the algorithm.

The framework of EMPPO is shown in Figure 4. Within the EMPPO framework, action masking is critical in addressing the reliability constraint of low-latency MEC networks. Two main enhancements reflect EMPPO's key advancements over the standard PPO. Firstly, EMPPO integrates a dynamic mask layer before the policy network's output to evaluate and filter potential actions based on current conditions and ensure compliance with error probability constraints arising from variable CSI and packet sizes. Secondly, during forward propagation, EMPPO incorporates an additional step where the mask is actively applied to refine the action probability distribution, ensuring that only feasible actions are considered in decision-making processes.

In practice, EMPPO converts binary-encoded action vectors into one-hot encoded vectors, representing a unique mapping for all potential UE selections. Under the FBL regime, the mask function comes into play by selectively filtering actions according to real-time CSI.

EMPPPO uses a clipped surrogate objective function to achieve a balance between exploitation and exploration, which can be expressed as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t [\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t)] \quad (20)$$

where θ is the policy network parameter, $r_t(\theta)$ is the probability ratio of action, \hat{A}_t is the advantage estimate at timestep t , and ϵ is a hyperparameter.

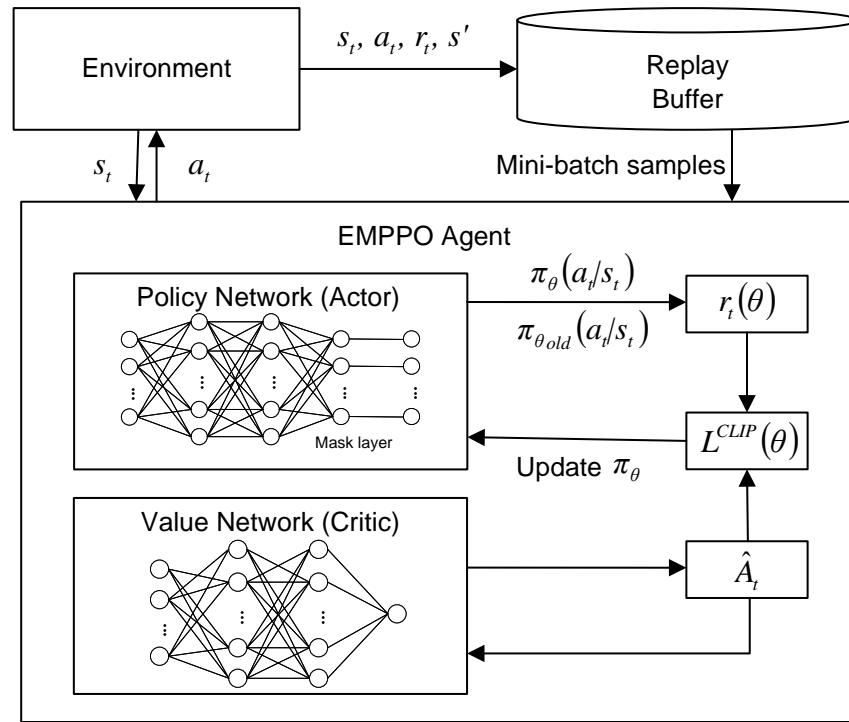


Figure 4. EMPPO framework.

The probability ratio $r_t(\theta)$ can be denoted as:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} \quad (21)$$

where $\pi_{\theta}(a_t | s_t)$ is the current policy, $\pi_{\theta_{old}}(a_t | s_t)$ is the old policy. If $r_t(\theta) > 1$, the agent is more likely to take action based on the current policy. If $0 < r_t(\theta) < 1$, the agent is more likely to take action based on the old policy.

The value network predicts the expected return from a given state, and the policy network outputs a probability distribution over actions given the current state. Generalized Advantage Estimation (GAE) is a method often used in PPO to reduce the variance of advantage estimates without greatly increasing bias, improving the stability of policy updates. The advantage function \hat{A}_t can be calculated as:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (22)$$

where r_t is reward, γ is the discount factor for future rewards and λ is GAE parameter. The temporal difference errors δ_t can be calculated as:

$$\delta_t = r_t + \gamma V_{\phi}(s_{t+1}) - V_{\phi}(s_t) \quad (23)$$

where $V_{\phi}(s_{t+1})$ is the value of the subsequent state, $V_{\phi}(s_t)$ is the value of the current state.

We employ an experience buffer for optimization. The agent's experience is stored as tuples (s, a, r, s') in an experience buffer and is then processed in epochs using mini-batches to improve policy parameters iteratively.

Based on the discussion, we can deploy the EMPPO algorithm in low-latency MEC environments. The specific steps are shown in Algorithm 1.

Algorithm 1 EMPPO algorithm for UE selection in low-latency MEC network.

Require: Initialize policy network $\pi_\theta(a|s)$ with parameters θ
Require: Initialize value function $V_\phi(s)$ with parameters ϕ
Require: Initial policy parameters $\theta_{old} \leftarrow \theta$
Require: Hyperparameters: clipping parameter ϵ , discount factor γ , GAE parameter λ

- 1: Initialize experience replay buffer \mathcal{B}
- 2: **for** $iteration \leftarrow 1$ to N **do**
- 3: Collect a set of partial trajectories $\mathcal{D}_{iteration}$ by running policy $\pi_{\theta_{old}}$
- 4: **for** each timestep t within these trajectories **do**
- 5: Compute advantage estimates \hat{A}_t using GAE
- 6: Store transition (s_t, a_t, r_t, s') in buffer \mathcal{B}
- 7: **end for**
- 8: **for** each epoch **do**
- 9: Sample mini-batch from buffer \mathcal{B}
- 10: **for** each sampled transition (s, a, r, s') **do**
- 11: Estimate probability ratio $r_t(\theta)$ using the current and old policies
- 12: Mask actions using function $M(s)$
- 13: **end for**
- 14: Optimize policy using the clipped objective function $L^{CLIP}(\theta)$
- 15: Update θ
- 16: **end for**
- 17: Update old policy parameters $\theta_{old} \leftarrow \theta$
- 18: **end for**

4. Simulation Results

In this section, we evaluate the performance of the EMPPO algorithm within a simulated low-latency MEC environment, encompassing diverse scenarios such as random task arrivals, varying data packet sizes, and dynamic channel conditions. This assessment aims to show the adaptability of EMPPO in real-time offloading decisions against the baseline strategies.

We developed our low-latency MEC network simulation environment in Python 3.9 and refined the algorithms from the widely recognized open-source Stable Baselines3 reinforcement learning library [38] to formulate our EMPPO approach. The simulation experiments were conducted on a setup running Windows 11, comprising an AMD Ryzen 7 5800 CPU paired with 32 GB of system memory and an NVIDIA GeForce RTX 3070 Ti graphics card.

In particular, we consider a low-latency MEC network composed of three UEs and one edge server. The setting of specific parameters is based on the 3GPP protocol as much as possible to be close to the real scenario. The error probability constraint for packet transmission $\varepsilon_{max} = 0.001$. Tasks arrive randomly in each frame with a probability $p = 0.4$, and the size of the data packet $m_{k,i}$ arriving at each UE typically varies between 500 and 2500 bits. Each UE transmits data packets to the edge server through independent wireless channels. The average SNR of three channels may vary and are set as $[\bar{\gamma} - \Delta\bar{\gamma}, \bar{\gamma}, \bar{\gamma} + \Delta\bar{\gamma}]$, where $\bar{\gamma} = 10$ dB, $\Delta\bar{\gamma} = 3$ dB. The channel correlation coefficient ρ is set as 0.8. The length of each frame is 1 ms, comprising 1500 symbols, with 1000 symbols for transmission and 500 symbols for computation, and one symbol's duration is 66.7 μ s.

In our proposed EMPPO framework, there are two networks: an actor network and a critic network. Both have one input layer, two hidden layers, and one output layer. The hidden layers are made up of 64 neurons. The other parameter settings of our algorithm are shown in Table 1.

Table 1. EMPPO algorithm parameters.

Description	Symbol	Value
Number of iterations	N	1×10^6
Number of episodes	n_e	5000
Learning Rate	α	0.0003
Number of Steps	n_{steps}	2000
Batch Size	B	100
Discount Factor	γ	0.99
Clip Range	ρ	0.2
Exploration Rate	ϵ	1
Exploration Decay Rate	δ	0.995
GAE Factor	λ^{GAE}	0.95

To evaluate the proposed algorithm, we compare our algorithm with the following baselines.

- (1) Random Offloading (RO): Randomly select some UEs for data packet offloading.
- (2) Average Offloading (AO): Select UEs for data packet offloading according to a predefined sequence.
- (3) Proximal Policy Optimization (PPO): The standard PPO algorithm, which has the same parameters as EMPPO.
- (4) Verified Random Offloading (VRO): Randomly select some UEs to offload data packets and verify that all selected packets within a frame can meet the error probability constraints.
- (5) Freshness Greedy Offloading (FGO): Always select the UE with the oldest packet for transmission, until the remaining blocklength makes the selected packet unable to meet the error probability constraints.
- (6) Channel Greedy Offloading (CGO): Always select the UE with best channel state for transmission, until the remaining blocklength makes the selected packet unable to meet the error probability constraints.

To minimize the impact of randomness, our proposed method and all baselines will be tested 500 times, and the average result of these tests will be taken as the output.

Figure 5 shows the convergence performance of the proposed EMPPO and the standard PPO. EMPPO starts its learning process with an average reward of around 50, higher than PPO's starting point of around 38. This is because EMPPO masks actions that can not meet the error probability constraints. Therefore, even when the action selection tends to be random at the beginning of the iteration, EMPPO can still achieve a better reward.

As the iterations progress, both EMPPO and PPO quickly improve their rewards in the first 500 episodes. After 500 episodes, the reward of EMPPO gradually converges to around 87, while the reward of PPO converges to about 45. This shows that EMPPO has a similar iteration efficacy to the standard PPO and can obtain a higher reward.

Figure 6 shows the influence of the task arrival probability. It can be seen that EMPPO has the lowest average task AuD across all task arrival probabilities, followed by the greedy methods FGO and CGO, followed by PPO, with RO constantly being the worst. As the task arrival probability increases, the average task AuD for all methods increases. This is due to an increased task arrival probability, potentially causing data packets to wait in the queue longer before being transmitted, and tasks have to wait longer for all the data packets to arrive. In this case, the average task AuD for CGO exceeds FGO's, which means freshness-based packet transmission (i.e., FGO) is more efficient than channel-based transmission (i.e., CGO). This is because CGO tends to send all the packets from channels with good channel quality first, making it harder to transmit packets from channels with poor channel quality. There is a certain gap between the performance of PPO and EMPPO, and the gap gradually increases with the increase in task arrival probability. This indicates that EMPPO can deal with a large number of data packets more effectively than PPO,

which may be because EMPPO reduces the exploration of invalid action space through variable masks.

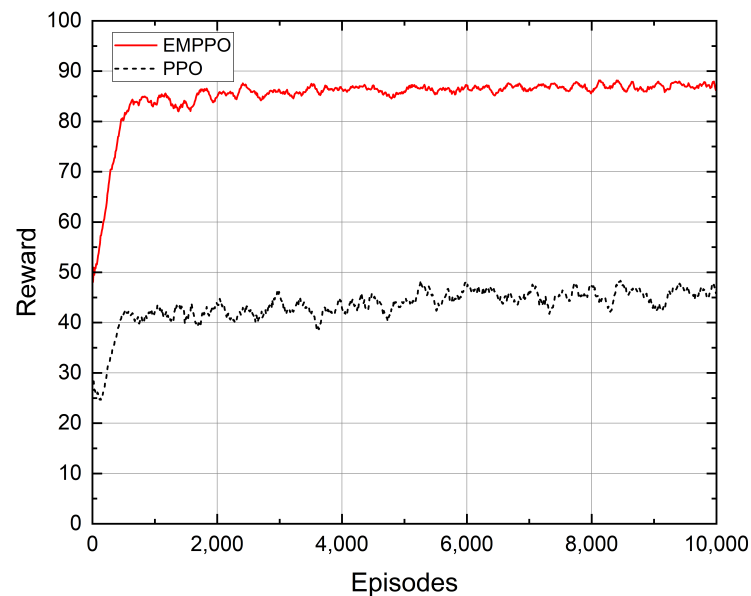


Figure 5. The convergence of reward.

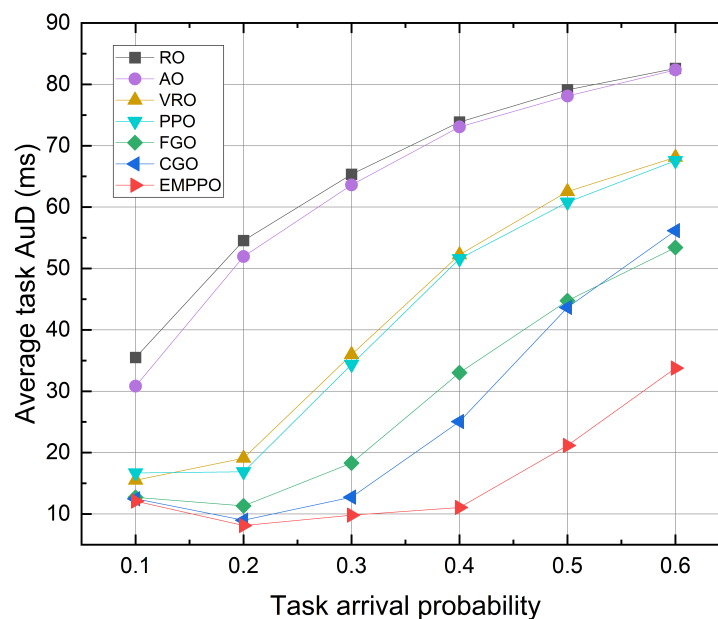


Figure 6. The average task AuD under different task arrival probability.

At low task arrival probability, EMPPO's performance is comparable to that of FGO and CGO, and the average task AuD initially decreases and then increases as the probability rises. This can be attributed to the less frequent arrival of new tasks and, as a result, a decelerated refresh rate of the task AuD. At high task arrival probability, EMPPO significantly reduces the average task AuD more than other methods. This suggests that EMPPO can successfully balance packet freshness and channel conditions, performing better under high-load scenarios.

Figure 7 shows the influence of the channel correlation coefficient. EMPPO consistently achieves the lowest average task AuD through various channel correlation coefficients. The average task AuD time for all methods exhibits a slow increase following a rise in the channel correlation coefficient. Among these, the greedy methods FGO and CGO show a greater sensitivity to changes in the channel correlation coefficient. A significant increase

in average task arrival time is observed under the extreme channel correlation coefficient $\rho = 0.99$. This is because the channel conditions change slowly at this channel correlation coefficient, making it difficult to take advantage of channel variations and transmit data packets with shorter blocklength during optimal channel conditions.

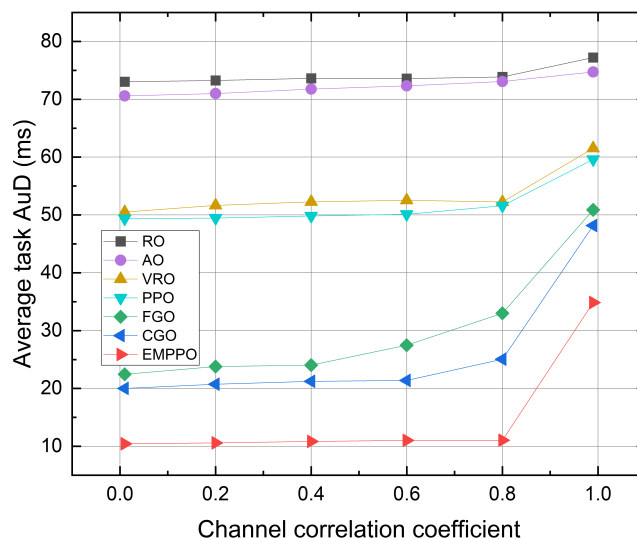


Figure 7. The average task AuD under different channel correlation coefficients.

Figure 8 shows the influence of the average SNR difference between channels. As the difference increases, the average task AuD for all methods grows, with EMPPO recording the lowest average task AuD, followed by CGO. Also, the gap in average task AuD between EMPPO and CGO gradually increases. This is mainly because CGO selects the UE corresponding to the channel with the highest SNR for data transmission. When the difference is zero, the channels can be regarded as three independent channels with the same average SNR, having equal transmission probabilities for each UE. However, as the difference increases, the channel with the highest average SNR is more likely to transmit data, while the channel with the lowest average SNR is likely to accumulate data packets. Figure 8 also shows that PPO and EMPPO methods based on reinforcement learning are less affected by the increase in channel differences, indicating that the method based on reinforcement learning can adapt to and balance the transmission of channels under different channel states.

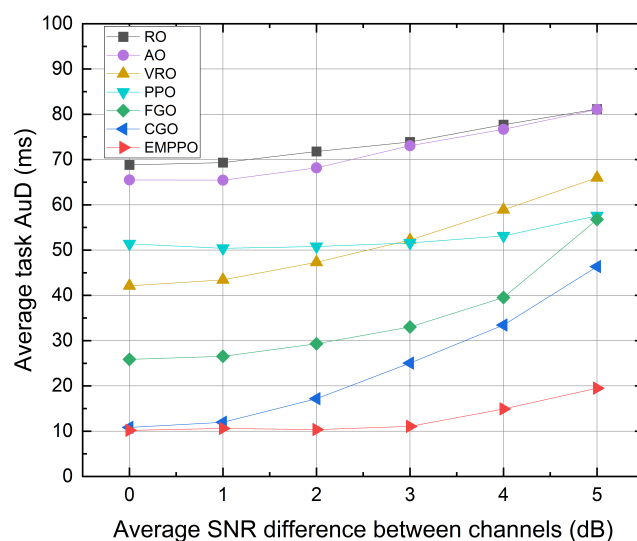


Figure 8. The average task AuD under different average SNR differences between channels.

Figure 9 shows the influence of the channel average SNR. As the SNR increases, the average task AuD of each method decreases. When the average SNR is high, data packets can be transmitted with a shorter blocklength, allowing more packets to be transmitted in a single frame. The results of methods based on greedy algorithms tend to be closer to EMPPO. However, when the average SNR drops, data packets require longer blocklength for transmission, and often, a task's data packets could be transmitted over multiple frames. In this scenario, deciding which data packets are transmitted in which frames significantly impacts the average task AuD. EMPPO performs better than other methods, suggesting that EMPPO is more applicable in environments with a low SNR. In extremely low SNR environments, a frame's total blocklength might not be enough to transmit a single data packet. Thus, the average task AuD differences among different methods become less pronounced.

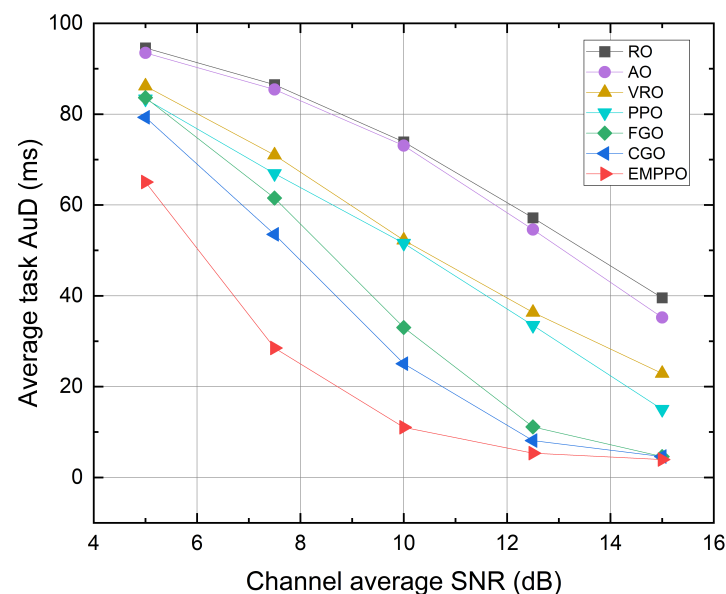


Figure 9. The average task AuD under different channel average SNR.

In Figure 9, we can observe that RO and AO show different convexity characteristics from other methods. That is because their performance approaches the maximum average task AuD. The maximum average task AuD is influenced by the number of simulation steps, which is set to 200 in the simulation. This leads to a maximum average task AuD of 100 ms (this corresponds to scenarios where no data packets are transmitted in any frame). An algorithm performing close to this upper bound will show a reduced rate of change, hence showing the distinct convexity. We can also observe this trend in Figures 6 and 10.

Figure 10 shows the influence of UE number. When the number of UEs increases, the task AuD of each method increases. This is because the one task's packet number is increasing (equal to the number of UEs). While the available blocklength for transmission is fixed, we need more frames to transmit the packets for one task. When the number of UEs is small, EMPPO's task AuD is similar to that of FGO and CGO. However, when the number of UEs is large, the performance of EMPPO is significantly better than other methods, which is consistent with the performance under different task arrival probabilities and confirms our analysis of Figure 6; that is, EMPPO can achieve better performance than other methods under high-load scenarios, and standard PPO has a lot of invalid action space to explore, which leads to a lower task AuD.

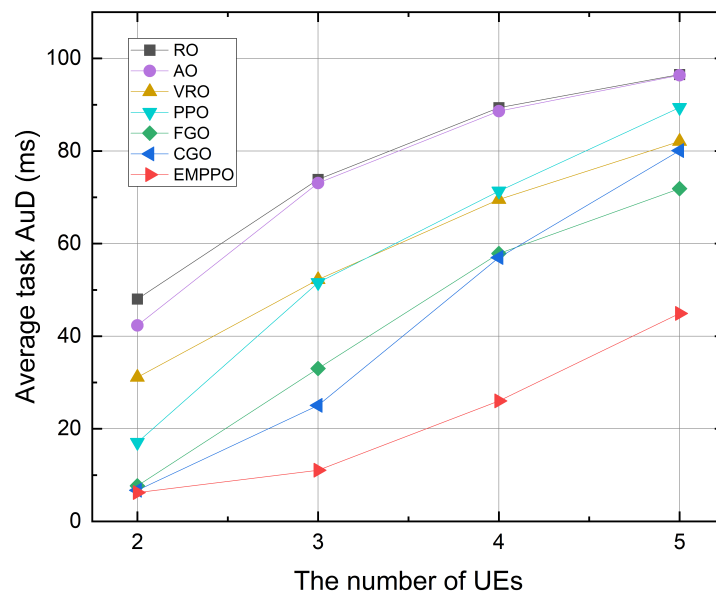


Figure 10. The average task AuD under a different number of UEs.

Figure 11 shows the influence of data packet size range. We have set up three different ranges for data packet size, each with the same average packet size of $m = 1500$ bits. Figure 11 shows that the average task AuD decreases as the packet size range decreases. Specifically, when the range is set to $[800, 2200]$, the average task AuD of EMPPO drops by about 33% and 7% compared to ranges $[300, 2800]$ and $[500, 2500]$. On the one hand, this is because a smaller packet size range results in a more concentrated distribution of packet size, which leads to a more focused distribution of the required blocklength, allowing the agent to predict the required blocklength for packets more accurately. On the other hand, the probability of encountering a large packet size is reduced, which decreases the probability of packets requiring a blocklength that exceeds frame length constraints. This enables packets to be transmitted successfully under relatively poor channel SNR.

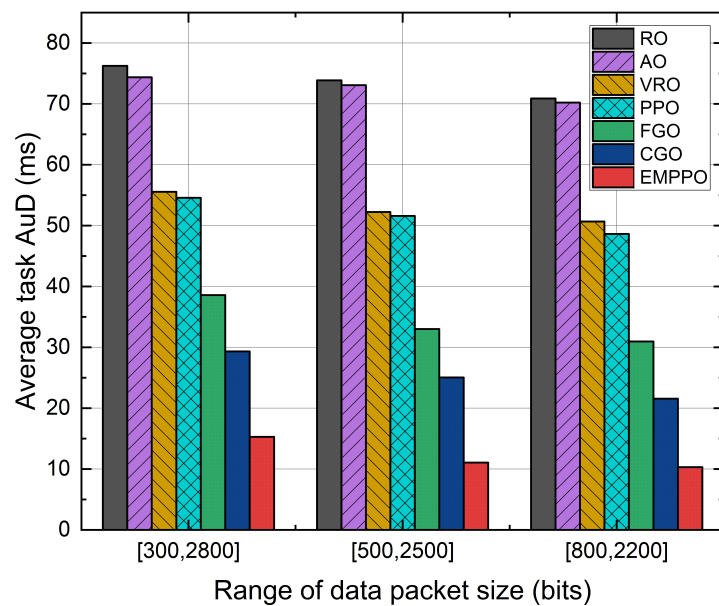


Figure 11. The average task AuD under different ranges of data packet size.

5. Conclusions

In this work, we proposed a latency optimization design for low-latency MEC networks. We optimized the average AuD for tasks through UEs selection and blocklength

allocation. Specifically, we considered random fading channels and packet transmission with FBL codes. We formulated an optimization problem to minimize the average task AuD under error probability constraints and, based on the FBL regime, transformed the joint optimization problem into an effective UE selection problem. Due to the dynamic task generation and random fading channels, the problem cannot be well solved analytically. We further proposed a DRL approach with the action-masked PPO method, which dynamically masks actions that cannot satisfy error probability constraints.

Simulation results validated the effectiveness of our proposed EMPPPO method. In particular, our method avoids exploring a large volume of the ineffective action space through the mask, significantly outperforming standard PPO methods. It also adapts better to scenarios with poor channel SNR and heavy task loads, reflecting the adaptability of DRL-based methods to dynamic environments. It is also worth mentioning that, although we assumed real-time CSI can be obtained, the proposed design can be extended to scenarios where only outdated or imprecise CSI is available, which is more realistic. Under such conditions, strict error probability constraints cannot be enforced, representing an exciting direction for our future work.

Author Contributions: Z.J. designed the original idea, performed the simulations, and drafted the manuscript. J.Y. and X.G. gave some suggestions and reviewed the manuscript. All authors have made substantive intellectual contributions to this study. All authors read and approved the final manuscript.

Funding: This research was supported by the Key Research and Development Project in the Hubei Province of China under grant No. 2022BCA035.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data generated or analyzed during this study are given in Section 4.

Acknowledgments: We would like to acknowledge all the subjects and technicians for their participation in and support of this study.

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

Abbreviations

The following abbreviations are used in this manuscript:

MEC	Mobile Edge Computing
FBL	Finite Blocklength
UE	User Equipment
AoI	Age of Information
AuD	Age upon Decision
PPO	Proximal Policy Optimization
IoT	Internet of Things
URLLC	Ultra-Reliable Low-Latency Communications
MDP	Markov Decision Process
SNR	Signal-to-Noise Ratio
FIFO	First-In-First-Out
GAE	Generalized Advantage Estimation
EMPPPO	Error Probability-Controlled Action-Masked Proximal Policy Optimization

References

1. You, X.; Wang, C.X.; Huang, J.; Gao, X.; Zhang, Z.; Wang, M.; Huang, Y.; Zhang, C.; Jiang, Y.; Wang, J.; et al. Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts. *Sci. China Inf. Sci.* **2021**, *64*, 1–74.
2. Al-Ansi, A.; Al-Ansi, A.M.; Muthanna, A.; Elgendy, I.A.; Koucheryavy, A. Survey on intelligence edge computing in 6G: Characteristics, challenges, potential use cases, and market drivers. *Future Internet* **2021**, *13*, 118. [[CrossRef](#)]
3. Javed, F.; Khan, Z.A.; Rizwan, S.; Shahzadi, S.; Chaudhry, N.R.; Iqbal, M. A Novel Energy-Efficient Reservation System for Edge Computing in 6G Vehicular Ad Hoc Network. *Sensors* **2023**, *23*, 5817. [[CrossRef](#)]

4. Peng, J.; Hou, Y.; Xu, H.; Li, T. Dynamic visual SLAM and MEC technologies for B5G: A comprehensive review. *Eurasip J. Wirel. Commun. Netw.* **2022**, 2022, 98. [\[CrossRef\]](#)
5. Van Huynh, D.; Nguyen, V.D.; Khosravirad, S.R.; Karagiannidis, G.K.; Duong, T.Q. Distributed Communication and Computation Resource Management for Digital Twin-Aided Edge Computing With Short-Packet Communications. *IEEE J. Sel. Areas Commun.* **2023**, 41, 3008–3021. [\[CrossRef\]](#)
6. Sabuj, S.R.; Asiedu, D.K.P.; Lee, K.J.; Jo, H.S. Delay optimization in mobile edge computing: Cognitive UAV-assisted eMBB and mMTC services. *IEEE Trans. Cogn. Commun. Netw.* **2022**, 8, 1019–1033. [\[CrossRef\]](#)
7. Kai, C.; Zhou, H.; Yi, Y.; Huang, W. Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability. *IEEE Trans. Cogn. Commun. Netw.* **2020**, 7, 624–634. [\[CrossRef\]](#)
8. Saleem, U.; Liu, Y.; Jangsher, S.; Li, Y.; Jiang, T. Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. *IEEE Trans. Wirel. Commun.* **2020**, 20, 360–374. [\[CrossRef\]](#)
9. Zhan, W.; Luo, C.; Min, G.; Wang, C.; Zhu, Q.; Duan, H. Mobility-aware multi-user offloading optimization for mobile edge computing. *IEEE Trans. Veh. Technol.* **2020**, 69, 3341–3356. [\[CrossRef\]](#)
10. Zhu, Y.; Hu, Y.; Yang, T.; Vogt, J.; Schmeink, A. Reliability-Optimal Offloading in Low-Latency Edge Computing Networks: Analytical and Reinforcement Learning Based Designs. *IEEE Trans. Veh. Technol.* **2021**, 70, 6058–6072. [\[CrossRef\]](#)
11. Liwang, M.; Gao, Z.; Wang, X. Let's trade in the future! A futures-enabled fast resource trading mechanism in edge computing-assisted UAV networks. *IEEE J. Sel. Areas Commun.* **2021**, 39, 3252–3270. [\[CrossRef\]](#)
12. Liwang, M.; Chen, R.; Wang, X.; Shen, X. Unifying futures and spot market: Overbooking-enabled resource trading in mobile edge networks. *IEEE Trans. Wirel. Commun.* **2022**, 21, 5467–5485. [\[CrossRef\]](#)
13. Polyanskiy, Y.; Poor, H.V.; Verdú, S. Channel coding rate in the finite blocklength regime. *IEEE Trans. Inf. Theory* **2010**, 56, 2307–2359. [\[CrossRef\]](#)
14. Merluzzi, M.; Di Lorenzo, P.; Barbarossa, S.; Frasca, V. Dynamic computation offloading in multi-access edge computing via ultra-reliable and low-latency communications. *IEEE Trans. Signal Inf. Process. Over Netw.* **2020**, 6, 342–356. [\[CrossRef\]](#)
15. Wu, Q.; Cui, M.; Zhang, G.; Wang, F.; Wu, Q.; Chu, X. Latency Minimization for UAV-Enabled URLLC-Based Mobile Edge Computing Systems. *IEEE Trans. Wirel. Commun.* **2024**, 23, 3298–3311. [\[CrossRef\]](#)
16. Huang, N.; Dou, C.; Wu, Y.; Qian, L.; Lin, B.; Zhou, H.; Shen, X. Mobile Edge Computing aided Integrated Sensing and Communication with Short-Packet Transmissions. *IEEE Trans. Wirel. Commun.* **2023**, 99, 1. [\[CrossRef\]](#)
17. Fu, Y.; Yang, X.; Yang, P.; Wong, A.K.; Shi, Z.; Wang, H.; Quek, T.Q. Energy-efficient offloading and resource allocation for mobile edge computing enabled mission-critical internet-of-things systems. *Eurasip J. Wirel. Commun. Netw.* **2021**, 2021, 26. [\[CrossRef\]](#)
18. Zhu, Y.; Yuan, X.; Hu, Y.; Wang, T.; Gursoy, M.C.; Schmeink, A. Low-Latency Hybrid NOMA-TDMA: QoS-Driven Design Framework. *IEEE Trans. Wirel. Commun.* **2023**, 22, 3006–3021. [\[CrossRef\]](#)
19. Yuan, X.; Zhu, Y.; Hu, Y.; Jiang, H.; Shen, C.; Schmeink, A. Latency-Critical Downlink Multiple Access: A Hybrid Approach and Reliability Maximization. *IEEE Trans. Wirel. Commun.* **2023**, 21, 9261–9275. [\[CrossRef\]](#)
20. Zhu, Y.; Hu, Y.; Schmeink, A.; Gross, J. Energy minimization of mobile edge computing networks with HARQ in the finite blocklength regime. *IEEE Trans. Wirel. Commun.* **2022**, 21, 7105–7120. [\[CrossRef\]](#)
21. Yang, T.; Hu, Y.; Gursoy, M.C.; Schmeink, A.; Mathar, R. Deep reinforcement learning based resource allocation in low latency edge computing networks. In Proceedings of the 2018 15th International Symposium on Wireless Communication Systems (ISWCS), Lisbon, Portugal, 28–31 August 2018; pp. 1–5.
22. Kaul, S.; Yates, R.; Gruteser, M. Real-time status: How often should one update? In Proceedings of the 2012 IEEE INFOCOM, Orlando, FL, USA, 25–30 March 2012; pp. 2731–2735.
23. Sung, H.; Kim, M.; Lee, S.; Lee, J. Age of information analysis for finite blocklength regime in downlink cellular networks. *IEEE Wirel. Commun. Lett.* **2021**, 11, 683–687. [\[CrossRef\]](#)
24. Han, B.; Zhu, Y.; Jiang, Z.; Sun, M.; Schotten, H.D. Fairness for freshness: Optimal age of information based OFDMA scheduling with minimal knowledge. *IEEE Trans. Wirel. Commun.* **2021**, 20, 7903–7919. [\[CrossRef\]](#)
25. Abdel-Aziz, M.K.; Samarakoon, S.; Liu, C.F.; Bennis, M.; Saad, W. Optimized age of information tail for ultra-reliable low-latency communications in vehicular networks. *IEEE Trans. Commun.* **2019**, 68, 1911–1924. [\[CrossRef\]](#)
26. Cao, J.; Zhu, X.; Jiang, Y.; Wei, Z.; Sun, S. Information age-delay correlation and optimization with finite block length. *IEEE Trans. Commun.* **2021**, 69, 7236–7250. [\[CrossRef\]](#)
27. Liu, Y.; Chang, Z.; Min, G.; Mao, S. Average age of information in wireless powered mobile edge computing system. *IEEE Wirel. Commun. Lett.* **2022**, 11, 1585–1589. [\[CrossRef\]](#)
28. Zhu, J.; Gong, J.; Chen, X. Minimizing Age-of-Information with Joint Transmission and Computing Scheduling in Mobile Edge Computing. *IEEE Internet Things J.* **2024**, 11, 9444–9457. [\[CrossRef\]](#)
29. Jiang, Y.; Liu, J.; Humar, I.; Chen, M.; Alqahtani, S.A.; Hossain, M.S. Age of Information-Based Computation Offloading and Transmission Scheduling in Mobile Edge Computing-Enabled IoT Networks. *IEEE Internet Things J.* **2023**, 10, 19782–19794. [\[CrossRef\]](#)
30. Tang, Z.; Sun, Z.; Yang, N.; Zhou, X. Age of Information of Multi-User Mobile-Edge Computing Systems. *IEEE Open J. Commun. Soc.* **2023**, 4, 1600–1614. [\[CrossRef\]](#)
31. Dong, Y.; Chen, Z.; Liu, S.; Fan, P. Age of information upon decisions. In Proceedings of the 2018 IEEE 39th Sarnoff Symposium, Neajark, NJ, USA, 24–25 September 2018; pp. 1–5.

32. Dong, Y.; Chen, Z.; Liu, S.; Fan, P.; Letaief, K.B. Age-Upon-Decisions Minimizing Scheduling in Internet of Things: To Be Random or To Be Deterministic? *IEEE Internet Things J.* **2020**, *7*, 1081–1097. [[CrossRef](#)]
33. Bao, Z.; Dong, Y.; Chen, Z.; Fan, P.; Letaief, K.B. Age-Optimal Service and Decision Processes in Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 2826–2841. [[CrossRef](#)]
34. Bao, Z.; Hu, Y.; Sun, P.; Boukerche, A.; Schmeink, A. Average age upon decisions with truncated HARQ and optimization in the finite blocklength regime. *Comput. Commun.* **2023**, *209*, 387–401. [[CrossRef](#)]
35. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
36. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1889–1897.
37. Huang, S.; Ontañón, S. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. In Proceedings of the The International FLAIRS Conference, Jensen Beach, FL, USA, 15–18 May 2022; Volume 35.
38. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.