IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 April 2025, accepted 14 April 2025, date of publication 21 April 2025, date of current version 28 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3562618

RESEARCH ARTICLE

Quality of Experience Optimization for AR Service in an MEC Federation System

HUONG MAI DO[1], TUAN PHONG TRAN[2], AND MYUNGSIK YOO[2], (Member, IEEE)
[1]Department of Information Communication Convergence Technology, Soongsil University, Seoul 06978, South Korea
[2]School of Electronic Engineering, Soongsil University, Seoul 06978, South Korea

Corresponding author: Myungsik Yoo (myoo@ssu.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by Korean Government of the Ministry of Science and ICT (MSIT), South Korea, Development of Candidate Element Technology for Intelligent 6G Mobile Core Network under Grant RS-2022-II221015; and in part by MSIT through the Information Technology Research Center (ITRC) Support Program supervised by IITP under Grant IITP-2025-RS-2021-II212046.

ABSTRACT Augmented reality (AR) in the internet of things requires ultra-low latency, high-resolution video, and fairness in multi-user environments, which pose challenges for traditional cloud and edge computing. To address this shortcoming, we studied AR subtask offloading and resource allocation in a multi-hop, multi-access edge computing federation. Our approach improves the quality of experience (QoE) by optimizing video quality and reducing delay while ensuring fairness, which is modeled as the ratio between provided and required quality. Instead of sequential execution, we adopt parallel AR subtask dependency processing to minimize latency. We propose an improved deep deterministic policy gradient algorithm for efficient solution exploration. Additionally, we implement strict training process monitoring to optimize resource usage and ensure sustainability. Experiments demonstrate that our method improves QoE by nearly 8% compared with TD3 while cutting training time in half.

INDEX TERMS Multi-access edge computing, MEC federation, augmented reality, resource allocation, quality of experience, deep reinforcement learning.

I. INTRODUCTION

Augmented reality (AR) is the real-time integration of digital information within a client environment. Unlike virtual reality, which constructs entirely synthetic virtual environments, AR provides multi-sensory encounters by blending real-world elements with computer-generated content, encompassing visual, auditory, haptic, somatosensory, and olfactory sensations. Consequently, AR has been used in various disciplines, including the internet of things (IoT), entertainment, business, education, and telemedicine [1]. With the extensive development of AR services and the advent of 5G and next-generation networks, AR in video streaming offers a highly realistic and immersive viewing experience for clients. The tasks associated with AR applications require considerable computing resources and continuous real-time computing. These requirements are difficult to satisfy when using only the resources of AR devices, such as glasses, head-mounted displays, or mobile phones.

Typically, a cloud computing system helps clients process tasks with significant computational loads [2]. Considering client distance and congestion from excessive access, it is infeasible for the cloud to accommodate delay-sensitive AR tasks. To solve this problem, studies on multi-access edge computing (MEC)-based IoT systems [3], [4], [5] have moved computing tasks closer to clients. However, the single-MEC model raises the potential issue of workload imbalances. Some MEC resources face an overload of tasks from clients, whereas others are underutilized. The MEC federation system, which is illustrated in Fig. 1, is a key technological advance that can solve this problem [6].

Many studies [7], [8], [9], [10], [11], [12], [13] have oversimplified AR tasks by treating them as typical computing tasks (without a subtask model), whereas other studies [14], [15], [16], [17] have attempted to address this shortcoming

The associate editor coordinating the review of this manuscript and approving it for publication was Alessandro Floris.

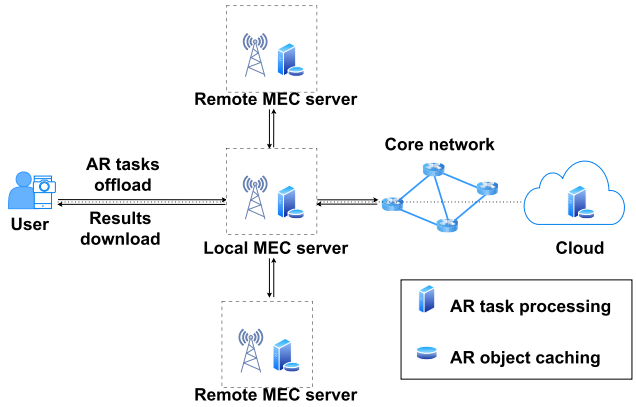

FIGURE 1. MEC collaboration-based AR system.

by dividing AR tasks into five subtask dependencies (video capture, tracking, mapping, object recognition, and rendering). However, this five-subtask model does not account for a key factor in AR information, namely the retrieval of AR objects. Furthermore, existing studies outline the sequential execution of subtasks [14], whereas this approach does not take advantage of the potential for optimization by parallelizing tasks based on their dependencies. For example, some subtasks can be executed concurrently to minimize latency or improve resource efficiency, which has been overlooked in previous studies. We attempt to address these issues by constructing an updated AR subtask dependency model with seven subtasks and a parallel working graph.

The optimization objective plays a vital role in optimizing the quality of experience (QoE). Previous studies have often focused on several main goals: reducing delays, reducing energy consumption, and improving video quality [7], [9], [11], [12], [13], [16], [17], [18]. However, existing methods optimize the total QoE (average of clients), which can unfairly prioritize certain clients to increase the total QoE. To address this imbalance, [8] raised the issue of fairness and proposed a solution that maximizes the minimum quality among clients. However, their approach is inefficient because each client's quality requirements are different (largely depending on the models of client devices and client requests). We propose a novel fairness model based on a function of the ratio between the quality of the received video stream and quality either requested or supported by devices.

Despite the proven effectiveness of machine learning (ML) methods for solving optimization problems, traditional algorithms, such as game-theoretic and search-based methods, still struggle with decision-making and scheduling problems, often leading to poor performance. Deep reinforcement learning (DRL), an advanced ML method, combines deep learning [19] and RL. DRL can provide high-quality solutions but consumes excessive time, computational resources, and a large amount of data. We applied the deep deterministic policy gradient (DDPG) algorithm to solve the target problem. To reduce training time, we enhanced the DDPG by considering multi-noise action (MNA) [20] and using a prioritized-experience replay (PER) buffer based on the weight of importance sampling (IS) [21]. MNA makes the action space more discoverable, whereas PER with IS helps accelerate and stabilize the training process.

Unlike computer vision ML tasks that typically output an accuracy value (up to 100%), the target of the reward function in DRL is a form of dynamic upper-bound optimization. Therefore, even when tracking the return of the reward in each training epoch, we cannot predict the distance to the maximum convergence value. As a result, setting a reward goal and expecting training to stop at this point is ineffective. To evaluate and compare algorithms while minimizing computational resource requirements, we designed a stopping condition for our experiments. Specifically, the experiments stopped when the algorithms reached convergence. We also logged real-time milestones to evaluate resource usage.

Aiming to address the issues outlined above, the main contributions of this study are as follows:

- We propose a practical AR application model that includes seven dependency subtasks, incorporates an *AR object retrieval* module and clearly defines the characteristics of each subtask to enable effective offloading decisions. We optimize subtask execution by exploring parallelization opportunities based on task dependencies rather than adhering to a strictly sequential workflow.
- We formulate a QoE optimization problem with a novel fairness metric for clients, aiming to maximize video quality and fairness while minimizing delay. This problem is transformed into a Markov decision process (MDP) for optimization.
- We propose an improved DDPG (IDDPG) algorithm to address the QoE optimization problem in large action spaces. An MEC system utilizing IDDPG enables more efficient decision-making for subtask offloading and resource allocation.
- We conduct extensive experiments with training tracking to demonstrate the effectiveness of the proposed framework. The results demonstrate that the proposed method outperforms the state-of-the-art method TD3 on AR video streaming, achieving nearly 8% better QoE while requiring only approximately half the training time.

The remainder of this paper is organized as follows. Existing studies on optimizing AR-based MEC systems are summarized in Section II. Section III discusses the proposed subtask model, system model, and objective function. In Section IV, we formulate an optimization framework under various resource and time constraints. Section V presents the IDDPG method used to solve the optimization problem. Section VI describes the simulation settings and results. A discussion of limitations is presented in Section VII. Finally, conclusions are drawn in Section VIII.

II. RELATED WORK

In the context of AR-based MEC systems, traditional MEC architectures with a single dedicated MEC server have been

studied extensively. Compared with such systems, an MEC federation enhances task performance and system efficiency. However, federation-based systems introduce additional complexity, requiring more advanced modeling and allocation strategies. Furthermore, AR tasks present unique characteristics and requirements that differ significantly from general IoT tasks, necessitating specialized approaches. Table 1 lists a summary of recent studies on AR task offloading and resource allocation within AR-based MEC systems. Table 2 delineates and compares our contributions with the existing literature.

A. DEDICATED MEC SYSTEMS

Dedicated MEC systems have a traditional MEC architecture with a single edge server that operates independently and only accesses the central cloud without collaborating with other edge servers. Most dedicated-MEC-based studies have not considered the AR subtask model issue [7], [8], [9], [10], [11] or have only considered an incomplete AR subtask dependency model [16].

1) NO SUBTASK MODEL

Cheng et al. [10] researched edge caching and computation in 5G for mobile AR and haptic internet, focusing on user power consumption and edge caching optimization under time constraints. They developed a technique for placing edge caches in the 5G network using a heuristic greedy algorithm and model selection approach to solve their optimization problem. Their focus was primarily on energy optimization, treating delay as a threshold without considering the specific processes and characteristics of AR tasks.

Li et al. [9] presented a collaborative model data computing service framework (CMCSF) for mobile web-based AR. The CMCSF creates a cooperative service that serves the MEC and cloud servers, along with a deployment plan to reduce delay. They further studied a collaborative scheduling strategy to improve the framework and adapt it to their optimized problem. Similar to [10], they also failed to address the complex model of AR subtasks.

Pan et al. [11] investigated the use of video-based artificial intelligence inside an MEC system. A mixed-integer nonlinear programming (MINLP) approach was devised to reduce inference delays and energy consumption while improving recognition accuracy. They presented a channel-aware heuristic approach for offloading decision optimization and used a distributed approach based on the alternate-direction method of multipliers. However, they limited the AR tasks to rendering only, leading to a less realistic optimization problem.

Zhang et al. [8] examined the provision of AR services in metaverse systems enabled by MEC. They aimed to solve two challenges, namely optimizing the QoE of users and resource allocation. They initially formulated a QoE model that factored in localization errors and then introduced a minimum-QoE maximization scheme to ensure user equity.

TABLE 1. Related works.

Reference	System model			AR task model		Optimization objective					Method	
	Dedicated MEC	MEC federation		No subtasks	Subtasks dependency	Latency	Energy	Frame quality	Bandwidth cost	Fairness	Traditional algorithm	ML based
		No multi-hop	Multi-hop									
[10]	✓			✓			✓				✓	
[7]	✓			✓		✓						✓
[9]	✓			✓		✓					✓	
[11]	✓			✓		✓	✓	✓			✓	
[8]	✓			✓				✓		✓	✓	
[16]	✓				✓	✓					✓	
[12]		✓		✓		✓					✓	
[17]		✓			✓	✓	✓				✓	
[18]		✓			✓	✓	✓		✓			✓
[14]		✓			✓	✓	✓					✓
[15]		✓			✓	✓	✓				✓	
Our work			✓		✓	✓		✓		✓		✓

TABLE 2. Our contributions.

Reference	F1	F2	F3	F4
[10]				
[7]				✓
[9]				
[11]				
[8]			✓	
[16]				
[12]				
[17]				
[18]				✓
[14]				✓
[15]				
Our work: AR-based MEC federation	✓	✓	✓	✓

F1: MEC federation using multi-hop connections
F2: AR subtask dependency model including AR object taking
F3: QoE optimization considering fairness
F4: Using DRL to solve the problem

However, this approach unbalanced the resources between users and caused unfairness when some users required more resources than others as a result of device heterogeneity.

Reference [7] studied a binary offloading approach for AR edge computing. This work focused on reducing latency while considering the computing power of mobile AR devices. The proposed offloading model, which was based on the DDPG-PER framework, allocated resources for beyond-fifth-generation networks. The inputs of this model were wireless channel gains and offloading states. However, this work only considered AR tasks as typical computing tasks while ignoring subtask dependencies.

2) SUBTASK DEPENDENCY MODEL

Unlike the studies mentioned above, [16] divided AR tasks into subtasks, including capturing, tracking, mapping, rendering, and objective recognition. An MEC-AR framework was established to leverage the capabilities of 5G cellular networks and enhance offloading decisions within a multi-tiered system. Regardless, the AR task model ignored the characteristics of each subtask. In reality, tasks, such as tracking, should be prioritized for processing on a local device or local server to ensure continuity and responsiveness. Offloading decisions were made based only on network and machine status. To reduce complexity, [16] relaxed the binary variables $\{0, 1\}$ of decisions to real values $[0, 1]$. However, this relaxation can lead to the loss of feasibility. Additionally, rounding a relaxed solution back to binary values can introduce rounding errors, further reducing the accuracy of the final solution.

B. MEC FEDERATION SYSTEM

In contrast to a dedicated MEC system, an MEC federation establishes a framework for orchestrating the movement of AR tasks across MEC servers within the system. Additionally, Xu et al. [12] did not consider the AR task model, whereas [14], [15], [17], [18] proposed AR subtask models and considered the dependencies among subtasks.

1) NO SUBTASK MODEL

Reference [12] aimed to optimize the service delivery engines for AR applications while enhancing the responsiveness of such applications within resource constraints. They constructed an online learning framework to maximize dynamic rewards without foreknowledge of the future influx of AR requests. In their network system, MEC servers were connected through the backhaul network. However, they assumed that the bandwidth in all connections was the same, meaning no bandwidth limitations existed. No guarantee exists that their online framework will work in real systems with significant backhaul congestion problems.

2) SUBTASK DEPENDENCY MODEL

Braud et al. [17] proposed a system model to enhance MEC's current task offloading model by offering multi-server, device-to-device (D2D), edge, and cloud offloading. Their model connected MEC servers through a backbone network and D2D through a multi-path configuration. They formulated an optimization function for total latency. A scheduling algorithm was applied to allocate six subtask dependencies for a multi-server and multi-device model under a multi-path routing network. However, they assumed a fixed additional latency for the connection between servers, which put the connection latency outside the scope of their optimization.

Peng et al. [15] investigated an AR application for healthcare cyber-physical systems (CPSs). They examined offloading AR application tasks in healthcare CPSs to edge computing while maintaining user privacy and mobility. They proposed a unique multi-objective meta-heuristic strategy based on the R2 indicator-II, which protected privacy while minimizing delay, load balancing, and energy consumption. However, they only considered a set of edge nodes without interconnection between them. The different AR subtasks could be offloaded to different edge nodes, but they assumed that the server handled most tasks, whereas user devices only handled video recording and display. Therefore, this model underestimated the capabilities of user devices.

In [18], the authors first described edge device performance issues and explained why dividing the AR overlay-rendering pipeline is necessary. They maximized user QoE while minimizing service costs, such as latency, bandwidth, and energy consumption. They developed a unique decision strategy based on DRL to address this challenge. However, the MEC servers were connected directly in pairs, and they treated all connections as equal, which is unrealistic. Additionally, they focused on only two subtasks (rendering and encoding), which they identified as heavyweight tasks.

Chen et al. [14] investigated an energy-efficient strategy for task offloading and resource allocation tailored to AR in

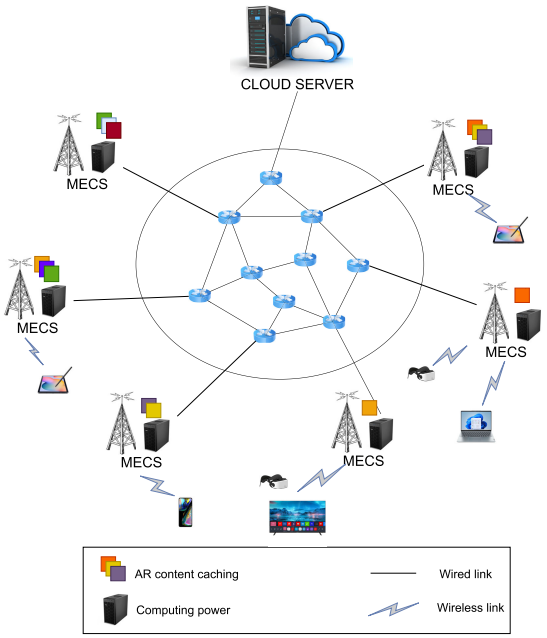

FIGURE 2. System model.

single- and multi-MEC systems. Initially, a model of the AR application was constructed, which manifested as a directed acyclic graph that reflected the inherent capabilities of the application. Similar to other works, this study lacked AR object-finding and processing components in the AR subtask model. The authors also assigned only the rendering subtask to the user equipment. The task offloading strategy was only applied to adjacent MECs, which narrowed the options for offloading. The authors proposed a strategy based on DRL in a dynamic communication context called the multi-agent DDPG framework.

In summary, based on the above analyses, AR-based MEC systems still face the following problems:

- Existing studies have not addressed the issue of subtask dependencies in AR services. Their subtasks are still scheduled within a narrow range (sequential).
- No existing studies have provided a comprehensive QoE optimization model that simultaneously considers video quality, fairness, and delay.
- Previous studies have considered MEC federation systems but neglected to clarify the interconnection among MEC federations.
- Previous studies used heuristic methods to solve optimization problems. Several studies applied DRL to the problem of resource allocation using many continuous variables. However, the use of DRL is challenging considering the very heavy training process required for a complex system, such as an MEC federation.

III. SYSTEM MODEL

As shown in Fig. 2, the system is composed of multiple smart AR devices in the set $\mathbb{N} = \{1, 2, \ldots, N\}$ in the bottom layer, MEC servers in the set $S_e = \{S_1, S_2, \ldots, S_E\}$ located with the base station (BS) in the middle layer, and a cloud server S_C in the top layer. AR clients may be smart glasses, head-mounted displays, smartphones, etc., which have low processing power and no AR content caching. Each MEC server contains ample computing resources. However, because the storage resources in each MEC server are limited, AR objects are distributed among the MEC servers, meaning that only a limited number of AR objects can be cached on one MEC server. We consider a scenario in which an AR client $n \in \mathbb{N}$ can run an AR application $m \in \mathbb{M}$ by offloading certain tasks to the servers or executing them locally, where $\mathbb{M} = \{1, 2, \ldots, M\}$ is a set of AR applications.

Additionally, an AR application model can be defined using subtask dependencies. Depending on the characteristics of each AR subtask and the system conditions, AR clients can offload a portion of their AR tasks to the MEC and cloud servers. The clients connect to the BSs on the MEC servers via wireless communication. The MEC servers can connect to nearby MEC servers or cloud servers through wired links. Routers are used for connections between servers. A multi-hop routing network allows servers to connect even when they are far apart. However, the infrastructure is considered to be preconfigured, and the servers are connected only through available links.

Therefore, we can define the set of all servers $\mathbb{S}$ as follows:

$$s \in \mathbb{S} = S_e \cup \{S_C\} = \{S_1, S_2, \ldots, S_E, S_C\}.$$

Each router $w \in \mathbb{R}$ and a set of routers $\mathbb{R}$ are defined as

$$w \in \mathbb{R} = \{R_1, R_2, \ldots, R_W\}.$$

Additionally, to determine the routing path from the client to the server, the following two factors are determined:

- The binary value $x_{n,s} \in \{0, 1\}$ denotes the connection status between client n and local MEC server s as

$$x_{n,s} = \begin{cases} 0, & \text{do not use connection client } n \text{ - server } s \\ 1, & \text{use connection client } n \text{ - server } s. \end{cases}$$

- The binary values $y_{n,w,w',k}, y_{n,w',w,k} \in \{0, 1\}$ denote a direct connection from router w to w'. The opposite direction w to w' is used to transfer the k^{th} subtask of client n. For $\{w, w'\} \subset \mathbb{R}$, the values are defined as follows:

$$y_{n,w,w',k} = \begin{cases} 0, & \text{do not use connection router } w \to w' \\ 1, & \text{use connection router } w \to w', \end{cases}$$

and

$$y_{n,w',w,k} = \begin{cases} 0, & \text{do not use connection router } w' \to w \\ 1, & \text{use connection router } w' \to w. \end{cases}$$

A. AR APPLICATION MODEL

In this section, we propose an AR application for seven subtasks corresponding to the recommended deadlines and computational resource requirements for each task. Unlike

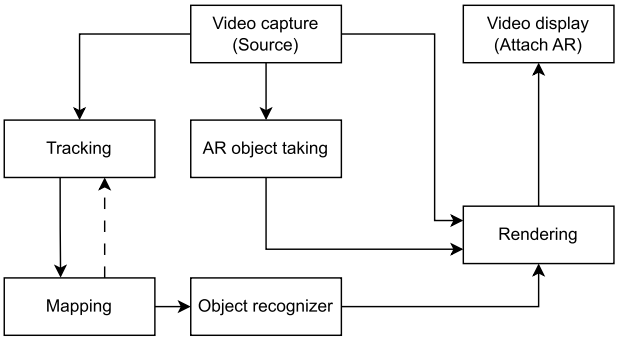

FIGURE 3. AR application model.

TABLE 3. AR-subtask data sizes.

Subtask definition	Input data size
Θ_n^1	I_n^1
Θ_n^2	$I_n^1\alpha_n^1$
Θ_n^3	$I_n^1\Pi_{k=1}^2\alpha_n^k$
Θ_n^4	$I_n^1\Pi_{k=1}^3\alpha_n^k$
Θ_n^5	$I_n^1\alpha_n^1$
Θ_n^6	$I_n^1(\alpha_n^1+\Pi_{k=1}^4\alpha_n^k+\alpha_n^1\alpha_n^5)$
Θ_n^7	$I_n^1\alpha_n^7(\alpha_n^1+\Pi_{k=1}^4\alpha_n^k+\alpha_n^1\alpha_n^5)$

the AR application model in [14] and [22], we add an *AR object taking* component considering the feasibility of distributing and storing AR objects in MEC servers. Individual MEC servers do not store all AR objects, as assumed in previous studies [7], [14], [16]. Fig. 3 presents this new AR application subtask model in detail.

Each AR application is divided into the following seven subtasks:

1) *Video capture* starts the process with a picture captured by a video source (data collected by the client's camera);
2) *Tracking* monitors the location of the clients within their surroundings;
3) *Mapping* builds a digital model and dynamic map of the surroundings;
4) *Object recognition* detects an identified item in the surrounding environment;
5) *AR object taking* stores and supplies AR objects;
6) *Rendering* combines other subtasks to prepare the processed frames with the virtual layer on top;
7) *Video display* displays the processed video (by monitor, mobile phone, or head-mounted display).

It is noteworthy that the *Video capture* and *Video display* components must be executed locally on client devices, whereas the *AR object taking* components can only be handled by the servers.

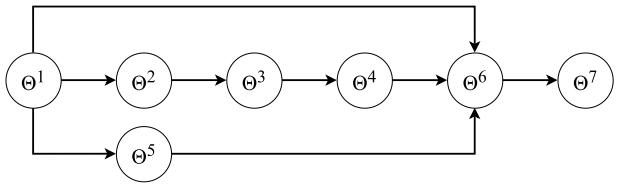

FIGURE 4. Subtask parallel dependency graph.

AR task processing requires significant computational resources, which can be attributed to the size of the tasks and process frequency. For example, an AR application allows the subsequent frame to be captured before the current frame has completed processing. Additionally, in each time slot t, one client n runs only one AR application m. $k \in \{1, 2, 3, 4, 5, 6, 7\}$ represents the kth subtask of the AR application. To describe the parameters of each subtask k of each client n, we define a tuple as follows:

$$\Theta_n^k = \{I_n^k, O_n^k, L_n^k, V_n\},$$

where I_n^k, O_n^k, and L_n^k denote the input AR subtask data size, result output size, and workload required to process a unit of data (CPU cycles/bit), respectively. V_n defines the highest resolution that the client device can support. $\alpha_n^k \in [0, 1]$ denotes the ratio of output to input of the subtask data size. Therefore, the data size of the output is determined as

$$O_n^k = I_n^k\alpha_n^k.$$

The seven subtask modules are *Capture*, *Tracking*, *Mapping*, *Object rec.*, *AR object taking*, *Rendering*, and *Display*. Θ_n^k identifies the subtask with $k \in \{1, 2, \ldots 7\}$. The types of input and output data for each subtask are also listed. Based on the dependency relationships between the subtasks, the data size calculation formulas are listed in Table 4. Therefore, given the size of the input data for the first subtask, the sizes of all subsequent subtasks may be determined, as listed in Table 3. Considering the dependencies between subtasks, the definitions of the subtasks are listed in Table 4.

Based on the characteristic properties of each subtask, the location that executes each subtask is listed in Table 5.

We extract a dependency subtask graph, as shown in Fig. 4. The dependency graph can be divided for parallel processing, even if each task depends on others. For example, tasks $\Theta^2 + \Theta^3 + \Theta^4$ and Θ^5 can be processed simultaneously. This model can improve the total latency by processing subtasks in parallel. A subtask does not have to wait for the previous subtask when it is not an input requirement.

Based on the definitions above, the AR-based MEC-federation workflow process follows the steps described below:

- First, AR devices capture videos at time step t, and the output of *Video capture* is the corresponding frames.
- Second, *Tracking* and *AR object taking* requests are generated. *Tracking* can be executed by AR devices or the local MEC server, whereas *AR object taking* can be

TABLE 4. AR-subtask definitions.

Subtask module	Capture	Tracking	Mapping	Object Rec.	AR object taking	Rendering	Display
Subtask definition	Θ_n^1	Θ_n^2	Θ_n^3	Θ_n^4	Θ_n^5	Θ_n^6	Θ_n^7
Data input	Request	Frame	Position	World Model	Frame	Frame Object property AR object	Rendered Frame
Input data size	I_n^1	$I_n^2 = O_n^1$	$I_n^3 = O_n^2$	$I_n^4 = O_n^3$	$I_n^5 = O_n^1$	$I_n^6 = O_n^1 + O_n^4 + O_n^5$	$I_n^7 = O_n^6$
Data output	Frame	Position	World Model	Object Property	AR Object	Rendered Frame	Frame Attached AR
Output data size	O_n^1	O_n^2	O_n^3	O_n^4	O_n^5	O_n^6	O_n^7

TABLE 5. AR-subtask execution locations.

	Capture	Tracking	Mapping	Object Rec.	AR object taking	Rendering	Display
Client device	✓	✓	✓	✓		✓	✓
Local server		✓	✓	✓	✓	✓	
Remote servers			✓	✓	✓	✓	
Cloud server			✓	✓	✓	✓	

handled by any MEC server that contains the objects or a cloud server.

- In the next steps, *Mapping* and *Object recognition* are executed. The requirement of *Mapping* results for *Tracking* is not a real-time process. The *Rendering* request is generated only when *Object recognition*, *AR object taking*, and *Video capture* are completed.
- After finishing *Rendering*, the frames attached to AR objects are displayed on the client's device.

B. DELAY MODEL

The delay model incorporates the following different types of delays: communication, computation, and queuing. We target an AR-based MEC federation network utilizing two distinct modes of communication for transmitting data (including tasks and results). The AR clients connect with a local MEC server via wireless connections, such as LTE/5G, whereas the MEC servers connect to neighbors and the cloud server through a wired network, such as optical fiber. We establish uplinks and downlinks for each type of connection. The uplink sends the AR task data, and the downlink sends the AR result data.

1) WIRELESS COMMUNICATION DELAY

Each client can only connect to the nearest MEC server, which is referred to as the local server, and the client must be within the local server's coverage area. The wireless channel gain between client n and server s is [23]

$$G_{n,s} = 127 + 30\log(dist_{n,s}),$$

where $dist_{n,s}$ is the distance between client n and local MEC server s, and R is the radius of the area covered by the local MEC server, which is defined as follows:

$$dist_{n,s} = \min_{s' \in S_e} dist_{n,s'} \leq R. \tag{1}$$

The uplink and downlink transmission rates can then be calculated using the method described in [23] as follows:

$$r_{n,s}^{up} = B_{n,s}^{up} \log_2\left(1 + \frac{P_n G_{n,s}}{N_0 B_{n,s}^{up}}\right), \tag{2}$$

$$r_{n,s}^{down} = B_{n,s}^{down} \log_2\left(1 + \frac{P_s G_{n,s}}{N_0 B_{n,s}^{down}}\right), \tag{3}$$

where $B_{n,s}^{up}$ and $B_{n,s}^{down}$ correspond to the uplink and downlink channel bandwidths allocated to these links, respectively; P_n and P_s indicate the transmission power allocated by client n and local MEC server s, respectively; and $N_0 = -174$ dBm/Hz is the power spectral density [24].

The wireless communication delay in the uplink and downlink, which are used for transferring subtask k of client n and the results, can be calculated as

$$d_{n,k}^{up} = \sum_{s \in S_e} \frac{I_n^k}{r_{n,s}^{up}} x_{n,s}, \tag{4}$$

$$d_{n,k}^{down} = \sum_{s \in S_e} \frac{O_n^k}{r_{n,s}^{down}} x_{n,s}. \tag{5}$$

The delay associated with the wireless connection of the uplink and downlink can be calculated as

$$d_{n,k} = d_{n,k}^{up} + d_{n,k}^{down}. \tag{6}$$

2) WIRED COMMUNICATION DELAY

The system connects its servers through wired networks. A backhaul network consists of a group of $\mathbb{R}$ connected

routers, as described previously. Therefore, a multi-hop connection transfers subtasks between servers in a network. Many other studies have assumed that the bandwidths of each wired link are equal throughout the entire system. However, this assumption is impractical. We present an in-depth investigation of the latency introduced during data transmission in multi-hop networks.

In contrast to the store-and-forward switching (SFS) method, cut-through switching (CTS) has the benefit of reducing latency because it does not require data to be stored at each hop before being forwarded [25]. The delay in a route consists of a data transmission delay across each hop and a backhaul delay proportional to the number of hops traversed. For example, let $t_{n,k}$ and N_{hop} represent the data transmission time of a single hop and the number of hops in the routing path, respectively. When SFS is utilized, the communication delay along the routing path is $t_{n,k}N_{hop}$. However, when employing CTS, this value is $t_{n,k} + \sigma N_{hop}$, where σ is a positive coefficient [26].

Let $\mathbb{P}^{up}_{n,k}$ and $\mathbb{P}^{down}_{n,k}$ represent the routing uplink and downlink paths for transferring the k^{th} subtask and result of client n, respectively. These routing paths are supported by R^{up}_P and R^{down}_P pairs of directly connected routers (w, w') and (w', w), which implies that $y_{n,w,w',k} = y_{n,w',w,k} = 1$. Because multiple routing paths share a direct uplink connection from router w to w', the throughput of the uplink channel (w, w') for transferring the k^{th} subtask and result of client n is calculated as follows:

$$\phi_{n,w,w',k} = \frac{B_{w,w'}}{\sum_{n\in N}\sum_{k\in K} y_{n,w,w',k}}, \tag{7}$$

where $\sum_{n\in N}\sum_{k\in K} y_{n,w,w',k}$ and $B_{w,w'}$ are the total number of subtask transmission channels for all users using connection (w, w') and total bandwidth of connection (w, w'), respectively. Similarly, the throughput of the downlink channel (w', w) is calculated as follows:

$$\phi_{n,w',w,k} = \frac{B_{w',w}}{\sum_{n\in N}\sum_{k\in K} y_{n,w',w,k}}. \tag{8}$$

The data transmission rate of the connection is determined by the end-to-end throughput, which is the lowest throughput among all links in the connection. Therefore, the throughput of the entire routing uplink and downlink path is determined as follows:

$$\phi_{\mathbb{P}^{up}_{n,k}} = \min\{\phi_{n,w,w',k} \forall (w, w') \in \mathbb{P}^{up}_{n,k}\}, \tag{9}$$

$$\phi_{\mathbb{P}^{down}_{n,k}} = \min\{\phi_{n,w',w,k} \forall (w', w) \in \mathbb{P}^{down}_{n,k}\}. \tag{10}$$

The data transmission latencies for routing paths $\mathbb{P}^{up}_{n,k}$ and $\mathbb{P}^{down}_{n,k}$ are calculated as follows:

$$t^{data,up}_{n,k} = \frac{I^k_n}{\phi_{\mathbb{P}^{up}_{n,k}}}, \tag{11}$$

$$t^{data,down}_{n,k} = \frac{O^k_n}{\phi_{\mathbb{P}^{down}_{n,k}}}, \tag{12}$$

where I^k_n and O^k_n denote the data input and output sizes, respectively.

The hop distance is directly proportional to the number of router pairs that the data traverses when utilizing CTS. Therefore, the wired communication delays of the uplink and downlink transmissions are calculated as follows:

$$t^{up}_{n,k} = t^{data,up}_{n,k} + \sigma(R^{up}_P + 1), \tag{13}$$

$$t^{down}_{n,k} = t^{data,down}_{n,k} + \sigma(R^{down}_P + 1), \tag{14}$$

where σ denotes the positive coefficient of each hop delay.

As a result, the wired communication delay associated with the wired connection of the uplink and downlink can be calculated as

$$t_{n,k} = t^{up}_{n,k} + t^{down}_{n,k}. \tag{15}$$

3) COMPUTATION DELAY

When a subtask is executed, a delay occurs during the computing process. In our proposed AR subtask model, all subtasks require computing resources, except for *AR object taking*. Therefore, we do not consider its computing delay. Instead, we consider the server containing the required AR object. Let $a_{n,s}$ define whether server s contains the required object for client n as

$$a_{n,s} = \begin{cases} 0, & \text{server } s \text{ does not contain object} \\ 1, & \text{server } s \text{ contains object.} \end{cases}$$

When $s = S_C$, the variable $a_{n,S_C} = 1$ indicates that the cloud server contains all required objects.

The binary value $z_{n,k,s} \in \{0, 1\}$ defines only one location to execute subtask k of client n as follows:

$$z_{n,k,s} = \begin{cases} 0, & \text{local device of client } n \text{ computes subtask} \\ 1, & \text{server } s \text{ computes subtask.} \end{cases}$$

The variables must satisfy $\sum_{s\in S}(z_{n,5,s}a_{n,s}) = 1$, which indicates that the server is used for subtask Θ^5_n because it contains the required object.

As mentioned previously, when $k = 5$, then $L^k_n = 0$. For the other subtasks, the number of CPU cycles required to process a single data unit is denoted as $L^k_n \neq 0$. Let $c_{n,k}$ represent the delay when executing subtask $k \in K$ for client n as

$$c_{n,k} = \sum_{s\in S} z_{n,k,s}\frac{I^k_n L^k_n}{f^k_{n,s}} + (1 - \sum_{s\in S} z_{n,k,s})\frac{I^k_n L^k_n}{f^k_n}, \tag{16}$$

where $f^k_{n,s}$ and f^k_n correspond to the allocated computing resources of server s or a local client device to execute subtask k of client n. L^k_n denotes the computing resources required for executing this subtask.

4) QUEUING DELAY

The server uses a buffer to store incoming unhandled tasks. This buffer is separate from the memory that stores the AR objects. The buffer operates on a first-in-first-out basis, and

the buffer capacity of each server is limited. If the buffer of a given server is full, it cannot receive any new subtasks.

Similar to the servers, each client device has a buffer that stores subtasks in a queue. The clients will not run out of memory because new tasks are only generated when they can store them. Based on the above definition, the queuing delay of subtask k from client n is calculated based on all previous subtask processing [27] as follows:

$$q_{n,k} = \sum_{s \in S} \sum_{n' \in \{N|n' \neq n\}} \sum_{k' \in \{K|k' \neq k\}} z_{n,k,s} c_{n',k'} + (1 - z_{n,k,s}) \sum_{k' \in \{K|k' \neq k\}} c_{n,k'},$$

where $c_{n',k'}$ denotes the computation delay of each previous subtask k' for client n in the buffer.

5) TOTAL DELAY

Based on the above formulations, the total end-to-end delay of each subtask k from client n is calculated as follows:

$$T_{n,k} = d_{n,k} + t_{n,k} + c_{n,k} + q_{n,k}. \tag{17}$$

However, the requirement of Θ_n^3 results in Θ_n^2 not being in real-time. Therefore, we can ignore the delay in the backhaul process. As mentioned in Section III-A, subtasks $\Theta_n^2 + \Theta_n^3 + \Theta_n^4$ and Θ_n^5 can be processed in parallel, starting with the output of subtask Θ_n^1 and ending with the input to subtask Θ_n^6. Therefore, the total delay in processing the AR application for each client is

$$T_n = \sum_{k=\{1,6,7\}} T_{n,k} + \max\{\sum_{k=2,3,4} T_{n,k}, T_{n,5}\}. \tag{18}$$

Maintaining an acceptable delay is critical to improving the QoE of AR applications. Therefore, the time limit of the AR application running on client n is set as T_n^{max}.

C. VIDEO QUALITY

In addition to delay, video quality is also an important factor affecting QoE. We presume that the reversed-difference mean opinion score (RDMOS) represents a downloaded video's quality [28]. The RDMOS ranges from 0 to 100, where better quality corresponds to higher values. The video resolution is represented by the data rate of the video played on client n, which is denoted as r_n. The mathematical representation of video quality may be expressed as a concave function of r_n [28] as follows:

$$Q_n = a \log(r_n) + b, \tag{19}$$

where a and b are 25 and -33, respectively [28].

D. FAIRNESS

Maximizing AR video stream quality in the shortest time is crucial for improving overall QoE in AR-based MEC systems. However, a simplistic approach can lead to unfair scheduling, benefiting overall QoE at the expense of certain clients. Fairness remains a critical factor in video streaming services [29].

Modern platforms, such as HTTP live streaming and dynamic adaptive streaming over HTTP, optimize streaming through adaptive bit rate mechanisms, selecting video resolutions based on user requests and network conditions. In addition to preventing resource waste (e.g., transmitting bit rates higher than device capabilities), our solution prioritizes fairness among users under resource constraints while striving for optimal video quality.

We further investigate QoE fairness to ensure reliable and fair video quality decisions for all clients. In our formulation, we maximize the fairness F_n by minimizing F_n^*, which is defined as the overall deviation of the ratio between the bit rate of video stream r_n and the highest video resolution that can support V_n for each client n as follows:

$$F_n = -F_n^* = -\left| \frac{r_n}{V_n} - \frac{1}{N-1} \sum_{n \in \{N|n' \neq n\}} \frac{r_{n'}}{V_{n'}} \right|. \tag{20}$$

IV. PROBLEM STATEMENT

We aim to maximize the average QoE of all clients by maximizing the video quality and fairness among clients while reducing delays. To obtain better video quality, a client must send a higher number of frames and pixels per frame, resulting in a large load and increasing the total end-to-end delay. Therefore, we should formulate a function to optimize the tradeoff between delay, video quality, and fairness.

To investigate the weighting parameters between the conflicting goals of our optimization problem, we must represent each weight as a multiple of two elements: the normalization value (α_1, β_1, γ_1) and weight of importance (w_Q, w_T, w_F) for each objective. QoE_n can be represented as follows:

$$QoE_n = \alpha_1.w_Q.Q_n - \beta_1.w_T.T_n + \gamma_1.w_F.F_n. \tag{21}$$

To avoid the effects of different units and magnitudes, we use three normalization factors to normalize the magnitude of each variable, which can be expressed as a division by the range distribution of each element as follows:

$$\alpha_1 = \frac{1}{Q^{max} - Q^{min}}, \quad \beta_1 = \frac{1}{T^{max} - T^{min}}, \quad \gamma_1 = \frac{1}{F^{max} - F^{min}}. \tag{22}$$

- Q_n: Video bit rate recommendation to ensure acceptable quality based on resolution and frame rate. In summary, the bit rate of an AR video should be higher than 500 Kbps (480p-SD), corresponding to the minimum RDMOS of 30 [28].
- T_n: The delay threshold of an AR application is more sensitive than that of a normal application. Therefore, [14] set the maximum delay to 25 ms.

TABLE 6. Weight & normalization values.

Items	Range	Nor. value (1 ÷ Range)	Weight	Final weight (Nor. value × Weight)
Q_n	$[30, 100]$	0.014286	70	1
T_n	$[0, 0.025]$	40	500	2×10^4
F_n	$[-1.5, 0]$	0.667	75	50

- F_n: As one of the first formulations of fairness, we set the range to $[-1.5, 0]$ to ensure fairness between users.
- $[Q^{min}, Q^{max}]$, $[T^{min}, T^{max}]$, $[F^{min}, F^{max}]$: The range distributions defining the lower and upper bounds for video quality, delay, and fairness, respectively.

We utilize weights to evaluate the importance of variables. The weights can be changed from zero to infinity depending on the target of optimization. In our setting, the values of the weights were set to {70, 500, 75}. The high delay weight w_T value stands out, demonstrating our optimization intent to focus on delay. Additionally, the smaller the value of w_Q, the better the video quality. A smaller w_F indicates better fairness. The values of the weights are listed in Table 6.

To balance video quality, end-to-end delays, and fairness, the objective function is expressed as follows:

$$\max_{x,y,z} QoE = \max_{x,y,z} \frac{1}{N} \sum_{n \in N} QoE_n. \tag{23}$$

We also adopt the following constraints:

$$C1: \quad \sum_{s \in S} x_{n,s} \leq 1, \tag{24}$$

$$C2: \quad \sum_{s \in S} z_{n,k,s} \leq 1, \tag{25}$$

$$C3: \quad \sum_{r \in \{S | r \neq s\}} y_{n,s,r,k} \leq 1, \quad \sum_{s \in \{S | s \neq r\}} y_{n,r,s,k} \leq 1, \tag{26}$$

$$C4: \quad \sum_{s \in S} (z_{n,5,s} a_{n,s}) = 1, \tag{27}$$

$$C5: \quad \sum_{n \in N} B^{up}_{n,s} \leq B^{up,max}_{s}, \quad \sum_{n \in N} B^{down}_{n,s} \leq B^{down,max}_{s}. \tag{28}$$

$$C6: \quad T_n \leq T^{max}_n, \tag{29}$$

$$C7: \quad \sum_{n \in N} \sum_{k \in K} f^k_{n,s} \leq F_s, \quad \sum_{k \in K} f^k_n \leq F_n, \tag{30}$$

$$C8: \quad \sum_{n \in N} \sum_{k \in K} z_{n,k,s} I^k_n \leq SE_s. \tag{31}$$

- $C1$ guarantees that each client n connects to at most one local MEC server s.
- $C2$ indicates that each task k of client n can be processed in only one location.
- $C3$ indicates that only one route can be chosen to transfer a subtask or result.
- $C4$ means that the selected server for handling the *AR object taking* subtask must contain the required AR object.
- $B^{up,max}_s$ and $B^{down,max}_s$ denote the maximum bandwidth of the uplink and downlink of each server s, respectively. The maximum bandwidth should not be exceeded by the uplink and downlink bandwidth allocated by each server, as indicated by $C5$.
- $C6$ states that the cumulative duration of each task cannot exceed the specified finish time.
- $C7$ specifies that the total computing resources allocated to all subtasks must not exceed the computing resource capacities F_s and F_n of server s and client n, respectively.
- $C8$ implies that the space used to store the subtasks in each server cannot exceed the server capacity SE_s.

System notations are listed in Table 7.

Similar to [23], the optimization problem described in Equation (23) includes the characteristics of a MINLP problem. The inherent computational complexity of MINLP problems arises from a combination of nonlinearity and integer variables when attempting to find an optimal solution under a sensitive delay. Therefore, we propose an IDDPG algorithm to address this optimization problem. The IDDPG algorithm uses PER-IS and MNA to enhance the original DDPG and makes decisions regarding subtask offloading and resource allocation to obtain optimal results.

V. METHODOLOGY

In this section, we describe the proposed task offloading and resource allocation framework for the AR-based MEC system. An MDP is established to model the optimization problem. Then, the DDPG approach is used to derive a solution to the MDP. However, the original DDPG method suffers from high complexity and a large action space. Therefore, we enhance it by considering MNA [20] and using a PER buffer based on the IS weight [21]. MNA makes the action space more discoverable, whereas PER with IS helps expedite and stabilize the training process.

A. MDP MODEL

1) STATE SPACE

At the beginning of time slot t, the prevailing state of the system environment is acquired by the agent. This information at timeslot t includes five sets as

$$s_t = \{K(t), I(t), F(t), A(t), SE(t)\},$$

where $K(t)$ represents the location of all subtasks in the system and $I(t)$ indicates their size. $F(t)$ denotes the computing resource state of all clients and servers in the

TABLE 7. Notation list.

Notation	Definition	Notation	Definition
$\mathbb{N}$	Set of AR clients	S_e	Set of MEC servers
$\mathbb{M}$	Set of AR applications	S_C	Cloud server
n	Client n	s, r	Servers s and r
w	Router	$\mathbb{R}$	Set of all network routers
V_n^k	Highest resolution client n can support	m	AR application m
Θ^k	Subtask k	$\mathbb{S}$	Set of servers
I^k	Subtask k input data size	$G_{n,s}$	Wireless channel gain between n and s
O^k	Subtask k output data size	$dist_{n,s}$	Distance between n and s
L^k	Workload required of subtask k	R	Server s coverage radius
α^k	Ratio of output and input subtask data	$r_{n,s}^{up}$	Wireless uplink transmission rate between n and s
T_n^{max}	Time to finish AR task in n	$r_{n,s}^{down}$	Wireless downlink transmission rate between n and s
$B_{n,s}^{up}$	Wireless uplink channel bandwidth between n and s	P_n	Transmission power allocated by n
$B_{n,s}^{down}$	Wireless downlink channel bandwidth between n and s	P_s	Transmission power allocated by s
N_0	Power spectral density	$d_{n,k}^{up}$	Delay of wireless uplink for subtask k of n
$x_{n,s}$	Wireless connection status between n and s	$d_{n,k}^{down}$	Delay of wireless downlink for subtask k of n
$y_{n,s,r,k}$	Direct wired connection status between s and r	$a_{n,s}$	AR object availability in MEC s
$z_{n,k,s}$	Location to execute the subtask	$c_{n,k}$	Delay for computing the subtask
ϕ	Throughput of directly wired link	$f_{n,s}^k$	Computing resource allocated by MEC s
$t_{n,k}^{up}$	Delay for wired communication in uplink	f_n^k	Computing resource allocated by device n
$t_{n,k}^{down}$	Delay for wired communication in downlink	$T_{n,k}$	Total end-to-end delay of subtask k of client n
$q_{n,k}$	Subtask queuing delay	T_n	Total end-to-end delay of the AR task of client n
r_n	Data rate of the video streamed to client n	QoE	Overall QoE of all clients
Q_n	Video quality for client n	$B_s^{up,max}$	Maximum uplink bandwidth of server s
F_n	Fairness of client n	$B_s^{down,max}$	Maximum downlink bandwidth of server s
F_s	Total computing resource (capacity) of server s	SE_s	Total storage (queuing) capacity of server s
F_n	Total computing resource (capacity) of client n	σ	Positive coefficient of each hop delay

system. $A(t)$ denotes the property of the servers, which is the AR object caching status, and $SE(t)$ denotes the MEC server storage resource status (for subtask queuing).

2) ACTION SPACE

Having evaluated the present state s_t, subsequent decisions $a(t)$ are then made by the agent, encompassing five distinct sets outlined as

$$a_t = \{f(t), B(t), x(t), y(t), z(t)\},$$

where $f(t)$ denotes the allocated computing resources of all clients and servers, $B(t)$ denotes the bandwidth assigned for all connections, and $x(t)$, $y(t)$, and $z(t)$ correspond to decisions regarding the wireless connections between clients and servers, directly wired connections between routers, and locations where AR subtasks are executed, respectively. The $x(t)$ and $y(t)$ actions correspond to routing path decisions.

DDPG requires actions to be continuous, but the sets $x(t)$, $y(t)$, and $z(t)$ are discrete. Therefore, we use the softmax function to rescale these values. The offloading and resource allocation decisions $x(t)$, $y(t)$, and $z(t)$ are represented by probabilities in the range $[0, 1]$, and their sum is one. The highest probability corresponds to a binary value of one in the above definition.

3) REWARD

The reward r_t serves as the environment's response to an agent executing an action a_t. In our specific scenario, the goal is to maximize the average QoE of all clients, and the reward is defined as the average QoE at each time step t. Therefore, the reward r_t is determined at each time step t as follows:

$$r_t = \frac{1}{N}\sum_{n\in N}(\alpha Q_n(t) - \beta T_n(t) + \gamma F_n(t)). \tag{32}$$

B. DDPG OVERVIEW

The DDPG model consists of four deep neural networks (DNNs), namely the actor network θ^μ, target actor network $\theta^{\mu'}$, critic network θ^Q, and target critic network $\theta^{Q'}$ [30]. The actor network aims to refine the policy $\mu(\theta^\mu)$, whereas the critic evaluates the quality of an action based on its associated Q value. The DDPG network employs two target networks as delayed versions, which help maintain and improve the original networks.

The value function $Q^\mu(s_t, a_t)$ is determined by each policy μ for the action–state pairings. This function quantifies the expected return from executing action a_t in the context of state s_t [23]. The approximations of the outputs from the actor and critic networks are as follows:

$$\mu(s_t|\theta^\mu) \approx \mu^*(s_t),$$
$$Q(s_t, a_t|\theta^Q) \approx Q(s_t, a_t).$$

Additionally, the output of θ^μ is augmented with Ornstein–Uhlenbeck (OU) random noise ϱ_t to enhance the exploration of actions [30] as follows:

$$a_t = \mu(s_t|\theta^\mu) + \varrho_t. \tag{33}$$

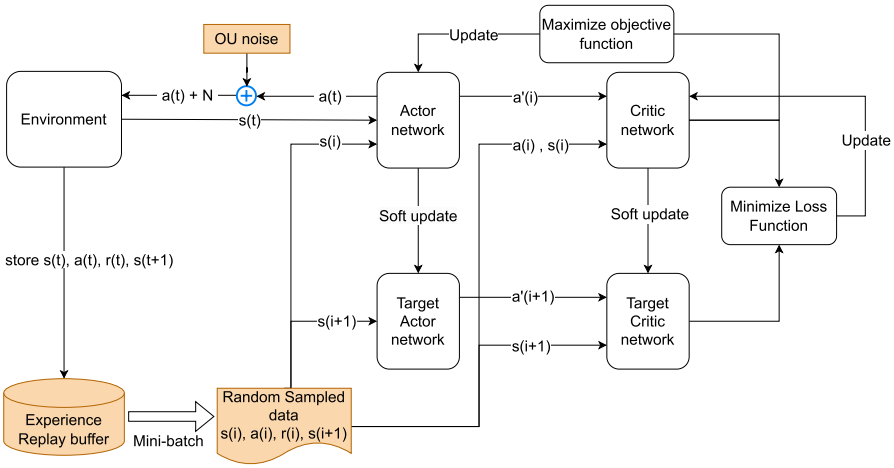

FIGURE 5. DDPG framework.

The training dataset is derived from the replay memory buffer R, which has a limited capacity of V. The tuple $\{s_t, a_t, r_t, s_{t+1}\}$ is saved in R for each time step t. The networks θ^{μ} and θ^{Q} update their weights by using randomly chosen mini-batches $\{s_i, a_i, r_i, s_{i+1}\}$ from R. The critic network uses the temporal difference (TD) error δ_i as its loss function, which can be derived as follows:

$$L(\theta^{Q}) = \frac{1}{M}\delta_i^2 = \frac{1}{M}\left(Q(s_i, a_i|\theta^{Q}) - y_i\right)^2, \tag{34}$$

where y_i and M denote the target Q value and data amount in each mini-batch, respectively. The loss function is minimized using the Adam optimizer [31] based on the following gradient equation:

$$\nabla_{\theta^{\mu}} J_{\mu *} = \frac{1}{M}\sum_{i=1}^{M} \nabla_{a_i}[Q^{\mu}(s_i, a_i|\theta^{Q}) \nabla_{\theta^{\mu}} \mu(s_i|\theta^{\mu})]. \tag{35}$$

The DDPG algorithm employs the current actor and critic network parameters, denoted as θ^{μ} and θ^{Q}, respectively, and performs soft updates as follows:

$$\begin{cases} \theta^{\mu'} \leftarrow (1-\delta)\theta^{\mu'} + \delta\theta^{\mu} \\ \theta^{Q'} \leftarrow (1-\delta)\theta^{Q'} + \delta\theta^{Q}. \end{cases} \tag{36}$$

The DDPG framework is illustrated in Fig. 5.

C. IMPROVED DDPG

In DRL frameworks, replay buffers are generally essential for maintaining model efficacy. Sampling from the buffer helps reduce data variability, prevents the loss of acquired knowledge, and enhances data diversity. Each sample within the replay buffer has a distinct impact on the system training process. As mentioned previously, DDPG employs a random selection process for mini-batches without considering their significance in the model. As a result, it consumes extensive training time because replaying meaningless samples does not result in large rewards. Therefore, we use the PER buffer [21], which utilizes prior knowledge to avoid strong correlations between incoming data.

Algorithm 1 IDDPG Algorithm (Using PER-IS and MNA)

Input: State space (environment state)
Output: QoE optimization, task-offloading and resource-allocation strategy, routing decision
Initialize critic $Q(s, a|\theta^{Q})$, weight θ^{Q}, and actor $\mu(\theta^{\mu})$, weight θ^{μ}
Initialize target critic Q', weight $\theta^{Q'} \leftarrow \theta^{Q}$ and target actor μ', weight $\theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer R, size V, and mini-batch M
Set priority $D_1 = 1$, exponents u, v
Initialize the number of episodes H';
for *episode* $= 1, H$ **do**
  Initialize K random process $\varrho_{\kappa}, \kappa \in \{1, 2, ..K\}$
  Initialize the system environment and obtain the initial observation state s_1
  for $t = 1, T$ **do**
    Generate K actions (40)
    for each action $\hat{a}_{t,\kappa}$ **do**
      Observe reward r_t and new state s_{t+1}
      Store experience $(s_t, \hat{a}_{t,\kappa}, r_t, s_{t+1})$ in R; set maximal priority $D_t = max_{i<t} D_i$
      if $t > V$ **then**
        for $j = 1, M$ **do**
          Sample experience j with probability (37)
          Compute IS weight (38)
          Set target Q value
          Update the priority of experience j according to the absolute TD error $|\delta_j|$
        end for
        Update critic network while minimizing loss function (34)
        Update actor network using policy gradient (35)
        Update target networks (36)
      end if
    end for
  end for
end for

Within the context of stochastic prioritization, the probability associated with sampling experience j is formulated as follows:

$$P(j) = \frac{D_j^{u}}{\sum_i D_i^{u}}, \tag{37}$$

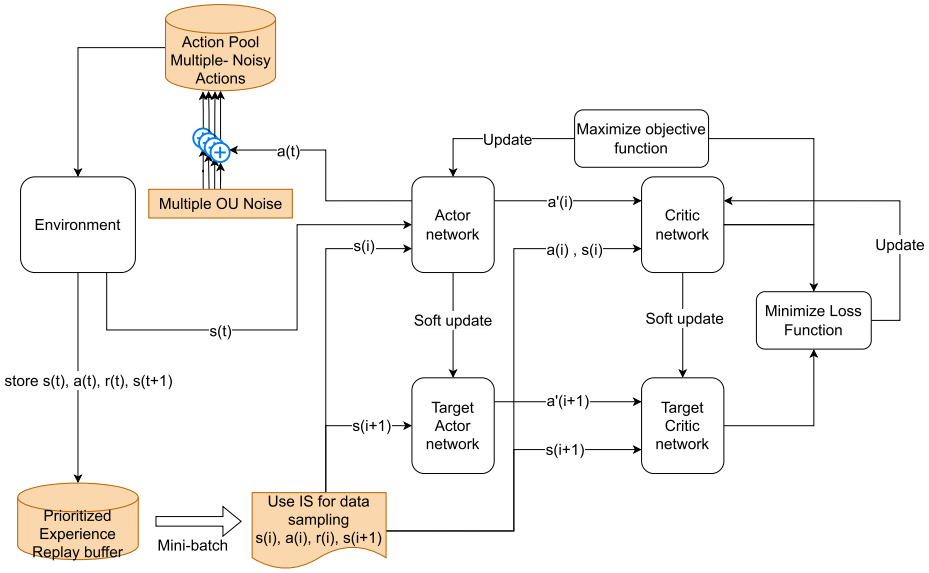

FIGURE 6. IDDPG (using PER-IS and MNA) framework.

where $rank(j)$ represents the position of experience j within R, whereas $D_j = \frac{1}{rank(j)} > 0$ represents the priority of experience j. We use the absolute TD error as a priority indicator to represent the importance of each experience.

Given that the TD error value δ inherently captures the extent of the learning potential within the samples, PER prioritizes the replay of experiences with the greatest TD error and executes a single action per iteration. However, incorporating PER into DDPG has two drawbacks:

- Experience samples with low TD error values in the first replays are often replayed infrequently or replayed only once, whereas cases with a high TD error are replayed repeatedly. This process leads to many valuable samples being missed, reduces diversity, and increases susceptibility to over-fitting.
- DDPG generates only one action per iteration. Our problem has a large action space with high complexity (multi-hop introduces a significant amount of routing options, especially when increasing the number of MEC servers), which extends the training process.

To address the first issue, we use the IS weight. This approach guarantees that the experiences with the lowest TD error are given a replay probability that is directly proportional to their priority. To address the second problem, our algorithm uses an MNA process. In contrast to the original DDPG, when operating the MNA process, multiple OU noises are added to the proto-action (output of the actor network) to create noisy actions and enhance exploration. Then, in each iteration, a set of actions is performed to increase exploration. The details of the PER-IS and MNA processes are presented below.

First, the IS weight is incorporated to mitigate the magnitude of the gradients and enhance training stability. The IS weight, as outlined in [32], is computed using the following formula:

$$w_j = \frac{1}{V^v.P(j)^v}, \tag{38}$$

where V represents the buffer size and v lies within the range [0, 1]. Furthermore, the weights are normalized by $\frac{1}{max_i w_i}$ to scale the updates downward effectively. Consequently, the differential loss function is defined as follows:

$$w_j \bigtriangledown_{\theta_1^Q} L(\theta_1^Q) = w_j.\delta_j. \bigtriangledown_{\theta_1^Q} .Q(s, a|\theta_1^Q). \tag{39}$$

By incorporating IS weights to expedite the training process, PER achieves a more comprehensive and dependable resolution of the issues encountered in DDPG.

Second, the DRL agent initializes K by adding OU noise processes to generate K noisy actions. The noisy action κ is $\hat{a}_{t,\kappa}$, which results from the combination of proto-action a_t and the κ^{th} OU noise as

$$\hat{a}_{t,\kappa} = a_t + \varrho_{t,\kappa}. \tag{40}$$

Therefore, with each proto-action a_t, the process can generate χ actions based on χ noises. The procedure for the IDDPG algorithm is defined in Algorithm 1 and illustrated in Fig. 6.

VI. EXPERIMENTS

This section presents the performance of the proposed framework for AR subtask offloading and resource allocation. The two goals of our experiments were collecting evidence of the effectiveness of an MEC federation compared with a dedicated MEC and collecting evidence of the effectiveness of the IDDPG method compared with other methods (when applied to our MEC federation model). Simulations were conducted on a personal computer running Windows 11

TABLE 8. Environment parameters.

Parameter	Value
AR client number	30
MEC server number	10
Cloud server number	1
Radius of each server-coverage region R	500 m [23]
Wireless link bandwidth (client–server)	$[0, 20]$ MHz [23]
Wired link transmission rate (routers)	$[0, 150]$ Mbps [23]
Transmission power of clients	15 dBm [14]
Transmission power of servers	30 dBm [14]
First subtask data size (size of frame)	$[15, 35]$ Kb [14]
Computing workload L (cycles/bit)	$[10, 1000]$ [14]
Client computing resources	$[0, 2]$ GHz [14]
MEC server computing resources	$[0, 20]$ GHz [14]
Cloud server computing resources	$[0, 200]$ GHz [33]
Delay threshold of AR application	25 ms [14]
Positive coefficient of CTS delay σ	0.0016 ms [34]
(a, b)	$(25, -33)$ [28]
(α, β, γ)	$(1, 2 \times 10^4, 50)$

TABLE 9. Model training parameters.

Parameter	Value
Sample batch size V	64
Mini-batch size M	64
Replay memory buffer size	10000
Maximum number of episodes	500
Steps in each episode	3000
Reward discount factor γ	0.9
Actor network learning rate	0.001
Critic network learning rate	0.0001
Soft replacement τ	0.01

(64-bit) with a 2.60 GHz Intel(R) Core(TM) i5-11400F CPU and 16 GB of RAM. The DRL implementation used Tkinter 8.6, Tensorflow 2.6, and Python 3.7. Anaconda was used as the development environment.

A. EXPERIMENTAL SETTINGS

The simulation scenario is as follows. One cloud server, 10 MEC servers, and 30 clients around the MEC servers are distributed in a 6 × 6 km area. An MEC server can connect to other servers or the cloud through multi-hop connections. 20 routers exist with at least one route between each pair of servers. Two routers within 2 km can connect directly to each other using a fiber cable. Each MEC server covers a 500-m radius and the clients inside that radius. If a client is allocated to an overlapping region with more than one MEC server, the client connects to the nearest MEC server. Table 8 lists all the parameters of our system.

The primary objective of the IDDPG model is to identify the optimal decisions for AR subtask offloading and resource allocation to obtain the maximum QoE. Our model consists of four five-layer networks. The IDDPG approach iteratively updates the model during system operation, allowing it to adapt to evolving environments dynamically. The batch and mini-batch sizes are set to 64. The training process begins by using the stored memory samples, and at each stage, the DNNs are trained to collect a significant amount of new data samples. DRL, which is a form of self-learning, generates an experience dataset through exploration. We define a dataset as a mini-batch regularly sampled from the replay buffer. Detailed information regarding model parameters can be found in Table 9.

B. PERFORMANCE EVALUATION

This study considers AR subtask offloading and resource allocation in an AR-based MEC federation. The proposed approach aims to optimize the overall QoE for all clients, which is essential for systems with limited resources. Consequently, we conducted experiments to assess the proposed model's performance in various settings.

The following approaches are compared:

- **Dedicated MEC–IDDPG (DM-IDDPG)**: Applying the IDDPG algorithm to a dedicated MEC server system, where each MEC server operates independently [7], [8], [9], [11], [16]. When MEC servers cannot execute subtasks, they can be sent to the central cloud.
- **MEC Federation–Greedy (MF-Greedy)**: The MEC servers collaborate or access the cloud server via multi-hop connections only when the current device is overloaded [10], [14]. Subtasks are executed successively on local clients and local servers until constraints are exceeded.
- **MEC Federation–TD3 (MF-TD3)**: Applying the state-of-the-art DRL framework TD3 to our multi-hop MEC federation system.
- **Proposed MEC Federation–IDDPG (MF-IDDPG)**: Applying the IDDPG algorithm to a multi-hop MEC federation system. Coordination between MEC servers is established to obtain optimal system performance.

To demonstrate the superiority of our AR subtask model compared with the conventional subtask model (sequential processing subtasks), we also performed the following experiments:

- **Sequential subtask model**: The system model is the same as our proposed MEC federation model, and the optimized method is IDDPG. However, the AR subtask model consists of seven subtasks that operate sequentially from *Capture* to *Display* without any parallel processing [14].
- **Our subtask model**: The system model is the same as our proposed MEC federation model, and the optimization method is IDDPG. Additionally, the AR subtask model consists of seven dependent subtasks with some parallel processing between subtasks depending on their characteristics.

1) PERFORMANCE COMPARISONS AND CONVERGENCE

First, we examine QoE metrics (calculated from the received video quality and delay) while following four approaches

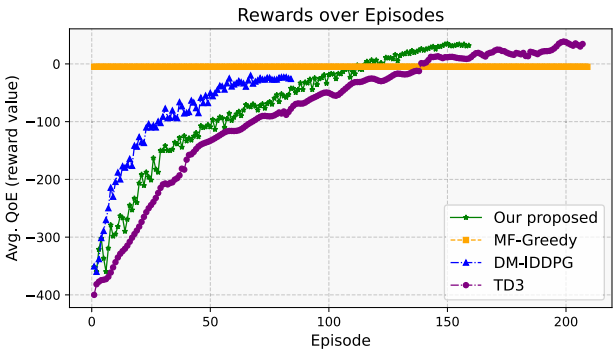

FIGURE 7. Average QoE comparison.

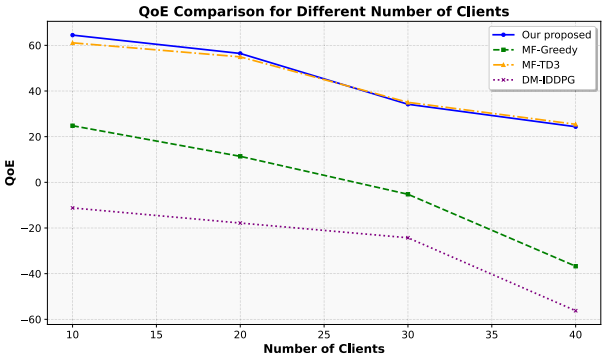

FIGURE 8. Impact of number of clients on QoE.

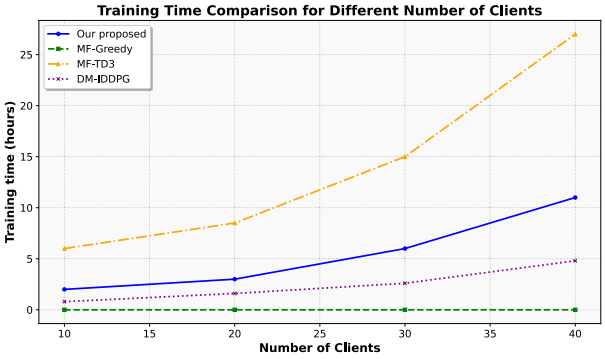

FIGURE 9. Impact of the number of clients on training time.

throughout the training process. The average QoE progression is presented in Fig. 7.

First, we compare MF-Greedy and the proposed MF-IDDPG, both of which contain similar system models but utilize different optimization strategies. Notably, the MF-Greedy technique is not learning-based. Using the same MF system, MF-Greedy provides a minimal QoE value of −5. The IDDPG approach helps the MF system achieve a QoE of 34. These results demonstrate that the proposed framework significantly improves network efficiency, particularly in complex environments, such as the MF environment.

We also evaluated the effectiveness of IDDPG in different systems. Fig. 7 reveals that the DM-IDDPG algorithm achieves a QoE value of −24, whereas the MF-IDDPG algorithm achieves a QoE value of 34. Improvements in the MF system are evidence of remarkable resource utilization and service quality optimization. Regardless, evidently, the rate at which IDDPG converges when implemented in the MF system is slightly lower than that in the DM system. DM-IDDPG exhibits the fastest convergence, converging after 74 episodes, whereas MF-IDDPG converges after 149 episodes. This result indicates that DM systems represent a far less complex and limited range of possible actions compared with MF systems. The MF system contains a significantly larger range of possible actions owing to the complex architecture of the multi-hop routing network. Consequently, longer processing times are required during the exploration phase.

Compared with MF-TD3, our method has a higher variance reward value. TD3 updates the policy network less frequently than the value network (i.e., the critics), which stabilizes learning and flattens the training reward curve. PER-IS focuses more on high-error experiences, which can make the learning curve spikier as the agent responds to the diverse priorities of sampled experiences. However, our convergence speed is faster than that of TD3 (episode 197, value 35.04) because of the aggressiveness of the PER-IS method in exploring priority values.

2) EFFECT OF INCREASING THE NUMBER OF CLIENTS

In this evaluation, we examine QoE results and training times for the four strategies with different numbers of clients (10, 20, 30, and 40). The number of MEC servers is set to 10, and the GPU capacity is set to 20 GHz, emphasizing that higher QoE values correspond to better performance. The results are presented in Fig. 8.

Overall, all strategies suffer reduced efficiency as the number of clients in the system increases. However, unlike the other strategies, MF-IDDPG maintained positive QoE values even with 40 clients. The QoE values of the 10, 20, 30, and 40 client cases are 64.48, 56.45, 34.19, and 24.35, respectively. Additionally, although the QoE degradation between 10 and 40 clients when using DM-IDDPG is equivalent to that of MF-IDDPG, the QoE values when using DM-IDDPG are negative for all four cases, indicating the ineffectiveness of the DM system.

TD3 employs a twin delayed network architecture, which necessitates a sufficiently large dataset to train the networks effectively during each iteration. Insufficient data can lead to suboptimal performance. This limitation explains why TD3 underperformed compared with our method in the scenario with only 10 clients.

In addition to QoE, an essential factor to consider when evaluating strategies employing DRL is training time. As shown in Fig. 9, the training time for the various strategies increases as the number of clients increases. MF-IDDPG requires training times ranging from 2 to 11 hours across different scenarios. Conversely, DM-IDDPG offers shorter

TABLE 10. Comparison of the average received video quality, delay, and fairness with different numbers of clients.

Objective	Number of clients	DM-IDDPG	MF-Greedy	MF-TD3	Proposed
Avg. video quality	10	91.25	94.59	97.59	99.22
	20	88.15	92.13	98.33	97.56
	30	86.72	90.11	96.28	95.43
	40	80.29	85.73	96.07	92.32
Avg. Delay (ms)	10	4.598	2.115	1.772	1.712
	20	4.799	2.512	2.114	2.013
	30	5	2.817	3.114	3.012
	40	6.001	3.611	3.415	3.316
Avg. Fairness	10	-0.21	-0.55	-0.021	-0.01
	20	-0.2	-0.61	-0.022	-0.017
	30	-0.22	-0.78	-0.025	-0.02
	40	-0.33	-1.005	-0.047	-0.033

training times (0.8, 1.6, 2.6, and 4.8 hours). Regardless, as stated previously, these strategies may not deliver optimal efficiency in terms of the QoE. Conversely, the MF-Greedy strategy, which does not rely on machine learning techniques, has a training time of 0 hours (the training time values overlap in the figure). Although this strategy may not require any training time, it fails to provide the desired performance levels necessary to address the optimization problem effectively.

The significant disadvantage of TD3 compared with our method is the training time and complexity of training. Running two neural networks in parallel and not selecting samples causes TD3 to spend excessive time on datasets with low contributions (from 6 to 27 hours). Additionally, the policy network updates less frequently than the critic network, causing delays in the updating of the optimization function. The training time increases exponentially with the increasing complexity of the system.

Table 10 lists the average values of the received video quality, delay, and fairness when the QoE is optimized using different approaches. As the number of clients increases, the video quality of the four approaches decreases. However, the values of the proposed MF-IDDPG approach are better than those of DM-IDDPG and MF-Greedy. The average delay of all methods is much lower than the time limit requirements, ranging from 1.712 to 6.001 ms. The approach that provides the highest average delay is DM-IDDPG, which occurs because a dedicated MEC system lacks resource allocation and cooperation between servers, causing the total task processing time to increase, particularly when many clients request a certain MEC. This factor also affects the fairness values of DM-IDDPG, which range from -0.33 to -0.2. In general, except for MF-Greedy, which produces the lowest average delay value with 30 clients, the MF-DDPG and MF-IDDPG methods have the best values for each metric when optimal QoE is achieved.

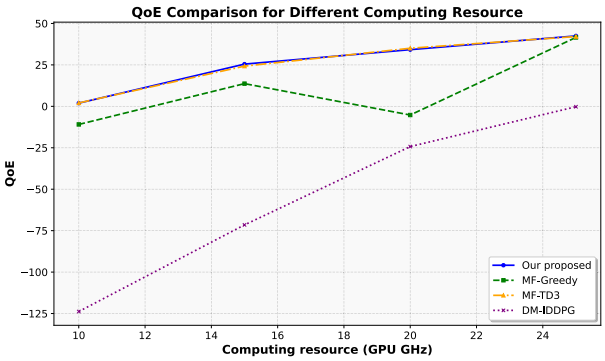

FIGURE 10. Impact of server computing resource capacity on QoE.

3) EFFECT OF INCREASING THE COMPUTING RESOURCES OF EACH SERVER

Fig. 10 presents the average QoE as the computing resource capacity of the servers increases. Compared with that of the other strategies, the QoE value of our proposed MF-IDDPG is the highest. The MF system with multi-hop connections can access the cloud and remote MEC servers, providing beneficial options. Additionally, each decision is handled by an optimal algorithm (IDDPG) that dynamically adjusts the strategy to ensure that client QoE is maximized while satisfying all constraints. In the initial stage (GPU = 10 GHz), the QoE of the proposed and MF-DDPG algorithms is the highest, followed by that of the MF-Greedy algorithm, and finally, the DM-IDDPG strategy. As the GPU resources increase, the QoE value of MF-Greedy increases and almost reaches the level of MF-IDDPG because a resource allocation

and task offloading algorithm are more effective in low-resource situations.

DM-IDDPG has the lowest QoE values when increasing the GPU resources, indicating that the DM system does not provide good performance, even when using a strong optimization algorithm, such as IDDPG, owing to the lack of load balancing for a large volume of tasks. The efficiency of the MEC system increases when the GPU resources of each DM server increase, as illustrated in Fig. 10 with a QoE increase of 123, which is three times that of the proposed method (increase of 41).

Table 11 lists the average values of the received video quality, latency, and fairness as the optimal QoE increases with different approaches. One can see that DM-IDDPG provides the lowest video quality values, followed by MF-Greedy. Using the same IDDPG optimization algorithm, the MF system performs better than the DM system when the computing resources of the servers increase. A comparison of MF-Greedy and MF-IDDPG reveals that learning-based methods outperform greedy methods.

In terms of the average delay, the lowest and most optimal delay values are provided by our proposed method when GPU $=$ 10 GHz and GPU $=$ 15 GHz. In the other two cases, the values of the proposed method are not minimal but are competitive, proving that the proposed method ensures optimal received video quality. However, to improve user QoE, latency must also be considered.

Similar trends can be observed for the average fairness value. In two out of four cases, MF-IDDPG produces the most optimal results. Changing the GPU does not cause large fluctuations in the sample dataset, so the methods based on DRL produce similar results.

4) AR SUBTASK MODEL COMPARISON

In this section, we compare our AR subtask model with a normal AR subtask sequence. The primary experimental parameters are listed in Table 8. The typical model executes seven subtasks in sequence, whereas our model allows *Tracking*, *Mapping*, and *Object recognition* to operate in parallel with *AR object taking*.

As listed in Table 12, the main difference between the results of the two subtask models is the average latency value. Parallelizing the subtasks significantly reduces the total execution time of a complete task. In theory, a longer processing time results in better video resolution. In the typical method, the video size does not change, but the processing time is prolonged, leading to a decrease in video quality (bit rate) compared with our model. Overall, based on the influence of the tradeoff weights, the values are magnified, leading to significant differences in QoE values between the two AR models.

5) SUBTASK OFFLOADING LOCATION

In each episode, 30 clients generate AR tasks. Each AR task is then divided into seven subtasks. After each episode, the algorithm exports information regarding subtask distribution

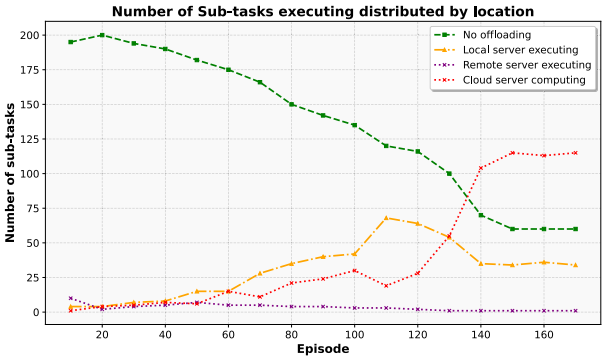

FIGURE 11. Subtask distribution changes over time.

(i.e., where the subtasks are executed, which can be the local client device, local MEC server, any remote MEC server, or cloud server).

The total number of subtasks for each episode is 210. We evaluate the distribution of the subtasks to be executed. Fig. 11 reveals that in the initial episodes, very few subtasks are offloaded (i.e., the subtasks are executed by local devices). However, as the number of training episodes increases, the number of offloaded operations increases and reaches the limit of 60. This result can be explained as follows.

When the number of subtasks executed by local devices is reduced, two types of subtasks, namely (*Video capture* and *Video display*), must still be performed by local devices. Only the first episodes send requests to the cloud server. Subsequent episodes only send one or two requests. The growth in the number of subtasks handled by the remote MEC server is evident after 110 episodes, proving that the proposed MEC federation system operates efficiently.

VII. DISCUSSION AND FUTURE WORK

Despite the promising results of our research, our current methodology has several limitations. A significant issue is the increased processing time required for handling dispersed AR content across multiple servers, which negatively affects overall system performance. Additionally, the training process for our model is lengthy and resource-intensive, posing challenges for practical deployment in real-world environments. Additionally, our work was limited to simulations, and integrating our model into a practical system requires additional resources. Furthermore, our methodology for weighting optimization variables remains at a fundamental stage, limiting its precision and applicability. Finally, certain assumptions made during the study, while necessary to simplify the modeling process, may influence the generalizability of our findings to diverse scenarios.

Based on the findings of this study, future work will address these limitations and explore several new directions. We will focus on optimizing the distribution process to reduce latency and explore alternative ML models that may improve system performance. Additionally, we aim to integrate

TABLE 11. Comparison of the average received video quality, delay, and fairness in servers with different GPU capacities.

Objective	GPU capacity	DM-IDDPG	MF-Greedy	MF-TD3	Proposed
Avg. video quality	**10**	72.25	75.62	82.01	82.65
	15	80.70	82.37	93.945	89.91
	20	86.72	90.11	96.28	95.43
	25	90.24	95.65	99.49	98.24
Avg. Delay (ms)	**10**	9.002	4.112	3.958	4.029
	15	7.011	3.221	3.471	3.215
	20	5	2.817	3.114	3.012
	25	3.999	2.521	2.789	2.743
Avg. Fairness	**10**	-0.32	-0.086	-0.016	-0.003
	15	-0.24	-0.0851	-0.0051	-0.0028
	20	-0.22	-0.78	-0.025	-0.02
	25	-0.21	-0.0732	-0.032	-0.018

TABLE 12. AR subtask model comparison.

Parameter Comparison	Our subtasks Model	Sequential Subtask model
Avg. QoE	**34.19**	26.52
Avg. Video quality	**95.43**	94.20
Avg. Delay (ms)	**3.012**	3.334
Avg. Fairness	**-0.02**	-0.02

sustainable development considerations by evaluating the tradeoffs between MEC system performance and energy consumption during the execution of DRL algorithms. Based on the outcomes of this study, we also plan to address user mobility concerns, which play a critical role in optimizing resource allocation within MEC federation systems. By incorporating mobility dynamics, we aim to enhance system performance and adaptability further in real-world scenarios.

VIII. CONCLUSION

In this paper, we proposed an MEC federation model for AR services. This framework is used to delegate computational AR tasks to MEC servers, and tasks are divided into seven subtasks. Rather than constraining subtask processing to local MEC servers, our approach allows these subtasks to be transferred to remote MEC servers through a multi-hop wired network, resulting in more efficient system resource utilization. Furthermore, we presented an effective decision-making procedure for resource allocation and task offloading, simplifying the implementation of real systems. This task was modeled as an MDP with realistic limitations. We proposed the MF-IDDPG algorithm for optimization. Extensive experiments demonstrated that the proposed MF-IDDPG framework outperforms competing systems. Additionally, we strictly monitored the training process to minimize the waste of computational resources. The results highlight the potential of our approach to improve resource allocation, reduce latency, and enhance overall system performance, making it a promising solution for real-world AR applications.

REFERENCES

[1] J. Xiong, E.-L. Hsiang, Z. He, T. Zhan, and S.-T. Wu, "Augmented reality and virtual reality displays: Emerging technologies and future perspectives," *Light, Sci. Appl.*, vol. 10, no. 1, p. 216, Oct. 2021.

[2] S. A. Bello, L. O. Oyedele, O. O. Akinade, M. Bilal, J. M. Davila Delgado, L. A. Akanbi, A. O. Ajayi, and H. A. Owolabi, "Cloud computing in construction industry: Use cases, benefits and challenges," *Autom. Construct.*, vol. 122, Feb. 2021, Art. no. 103441.

[3] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1160–1192, 2nd Quart., 2021.

[4] K. Zheng, G. Jiang, X. Liu, K. Chi, X. Yao, and J. Liu, "DRL-based offloading for computation delay minimization in wireless-powered multi-access edge computing," *IEEE Trans. Commun.*, vol. 71, no. 3, pp. 1755–1770, Mar. 2023.

[5] X. Liu, B. Xu, K. Zheng, and H. Zheng, "Throughput maximization of wireless-powered communication network with mobile access points," *IEEE Trans. Wireless Commun.*, vol. 22, no. 7, pp. 4401–4415, Jul. 2022.

[6] M. Suzuki, T. Joh, H. Lee, W. Featherstone, N. Sprecher, D. Sabella, N. Oliver, S. Shailendra, F. Granelli, C. Costa, L. Chen, H. Nieminen, O. Berzin, and F. Naim, "Mec federation: Deployment considerations," Eur. Telecommun. Standards Inst., Sophia Antipolis, France, White Paper 49, 2022.

[7] M. Chen, W. Liu, T. Wang, A. Liu, and Z. Zeng, "Edge intelligence computing for mobile augmented reality with deep reinforcement learning approach," *Comput. Netw.*, vol. 195, Aug. 2021, Art. no. 108186.
[8] H. Zhang, S. Mao, D. Niyato, and Z. Han, "Location-dependent augmented reality services in wireless edge-enabled metaverse systems," *IEEE Open J. Commun. Soc.*, vol. 4, pp. 171–183, 2023.
[9] L. Li, Q. Lu, Y. Xu, H. Zhang, and Y. Li, "CMCSF: A collaborative service framework for mobile Web augmented reality base on mobile edge computing," *Computing*, vol. 103, no. 10, pp. 2293–2318, Oct. 2021.
[10] Y. Cheng, "Edge caching and computing in 5G for mobile augmented reality and haptic Internet," *Comput. Commun.*, vol. 158, pp. 24–31, May 2020.
[11] G. Pan, H. Zhang, S. Xu, S. Zhang, and X. Chen, "Joint optimization of video-based AI inference tasks in MEC-assisted augmented reality systems," *IEEE Trans. Cognit. Commun. Netw.*, vol. 9, no. 2, pp. 479–493, Apr. 2023.
[12] Z. Xu, D. Liu, W. Liang, W. Xu, H. Dai, Q. Xia, and P. Zhou, "Online learning algorithms for offloading augmented reality requests with uncertain demands in MECs," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 1064–1074.
[13] V. Cozzolino, L. Tonetto, N. Mohan, A. Y. Ding, and J. Ott, "Nimbus: Towards latency-energy efficient task offloading for AR services," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1530–1545, Apr. 2022.
[14] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10843–10856, Jul. 2021.
[15] K. Peng, P. Liu, M. Bilal, X. Xu, and E. Prezioso, "Mobility and privacy-aware offloading of AR applications for healthcare cyber-physical systems in edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 2662–2673, Sep. 2022.
[16] A. Younis, B. Qiu, and D. Pompili, "Latency-aware hybrid edge cloud framework for mobile augmented reality applications," in *Proc. 17th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2020, pp. 1–9.
[17] T. Braud, P. Zhou, J. Kangasharju, and P. Hui, "Multipath computation offloading for mobile augmented reality," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2020, pp. 1–10.
[18] L. Zhang, X. Wu, F. Wang, A. Sun, L. Cui, and J. Liu, "Edge-based video stream generation for multi-party mobile augmented reality," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 409–422, Jan. 2024.
[19] H. A. Hoang and M. Yoo, "3ONet: 3-D detector for occluded object under obstructed conditions," *IEEE Sensors J.*, vol. 23, no. 16, pp. 18879–18892, Aug. 2023.
[20] P. T. A. Quang, Y. Hadjadj-Aoul, and A. Outtagarts, "A deep reinforcement learning approach for VNF forwarding graph embedding," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1318–1331, Dec. 2019.
[21] Y. Hou and Y. Zhang, "Improving DDPG via prioritized experience replay," Dept. Inf. Eng., Chinese Univ. Hong Kong, Shatin, Hong Kong, Tech. Rep., May 2019.
[22] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Netw.*, vol. 33, no. 4, pp. 162–169, Jul. 2019.
[23] H. Mai Do, T. P. Tran, and M. Yoo, "Deep reinforcement learning-based task offloading and resource allocation for industrial IoT in MEC federation system," *IEEE Access*, vol. 11, pp. 83150–83170, 2023.
[24] S. Yang, "A joint optimization scheme for task offloading and resource allocation based on edge computing in 5G communication networks," *Comput. Commun.*, vol. 160, pp. 759–768, Jul. 2020.
[25] A. Astarloa, J. Lázaro, U. Bidarte, J. A. Araujo, and N. Moreira, "Fpga implemented cut-through vs store-and-forward switches for reliable Ethernet networks," in *Proc. Design Circuits Integr. Syst.*, 2014, pp. 1–6.
[26] K. Shin, S. Choi, and H. Kim, "Flit scheduling for cut-through switching: Towards near-zero end-to-end latency," *IEEE Access*, vol. 7, pp. 66369–66383, 2019.
[27] F. Fu, Z. Zhang, F. R. Yu, and Q. Yan, "An actor-critic reinforcement learning-based resource management in mobile edge computing systems," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 8, pp. 1875–1889, Aug. 2020.
[28] C. Chen, X. Zhu, G. de Veciana, A. C. Bovik, and R. W. Heath Jr., "Rate adaptation and admission control for video transmission with subjective quality constraints," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 1, pp. 22–36, Feb. 2015.
[29] P.-Y. Chou, W.-Y. Chen, C.-Y. Wang, R.-H. Hwang, and W.-T. Chen, "Pricing-based deep reinforcement learning for live video streaming with joint user association and resource management in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 6, pp. 4310–4324, Jun. 2022.
[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[32] A. R. Mahmood, H. P. V. Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, Dec. 2014, pp. 3014–3022.
[33] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020.
[34] J. Woods, "Cut-through considerations and impacts to industrial networks," *Proc. IEEE*, vol. 802, pp. 1–18, May 2017.

HUONG MAI DO received the B.S. degree in electronics and telecommunications from Hanoi University of Science and Technology, Hanoi, Vietnam, in 2020, and the M.S. degree from Soongsil University, Seoul, South Korea. Her research interests include edge computing, resource allocation, and task scheduling.

TUAN PHONG TRAN received the B.S. degree in information technology from Le Quy Don Technical University, Vietnam, in 2019. He is currently pursuing the M.S. degree with the School of Electronic Engineering, Soongsil University, South Korea. His current research interests include machine learning, deep learning, and edge computing.

MYUNGSIK YOO (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, South Korea, in 1989 and 1991, respectively, and the Ph.D. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 2000. He was a Senior Research Engineer with the Nokia Research Center, Burlington, MA, USA. He is currently a full-time Professor with the School of Electronic Engineering, Soongsil University, Seoul. His research interests include visible-light communications, cloud computing, and machine learning.