Multiagent Deep Reinforcement Learning for Task Offloading and Resource Allocation in Cybertwin-Based Networks

Wenjing Hou, *Graduate Student Member, IEEE*, Hong Wen, *Senior Member, IEEE*, Huanhuan Song, *Graduate Student Member, IEEE*, Wenxin Lei, *Graduate Student Member, IEEE*, and Wei Zhang, *Fellow, IEEE*

***Abstract*—In this article, a hierarchical task offloading strategy is presented for delay-tolerant and delay-sensitive missions by integrating edge computing and artificial intelligence into Cybertwin-based network to guarantee user Quality of Experience (QoE), low latency, and ultrareliable services, which are huge challenges to the Internet of Things (IoT) due to diverse application requirements, heterogeneous multidimensional resources, and time-varying network environments. The novel scheme achieves faster task processing, dynamic real-time allocation, and lower overhead by taking advantages of a multiagent deep deterministic policy gradient (MADDPG). Moreover, federated learning is used to train the MADDPG model. Numerical results demonstrate that the proposed algorithm improves system processing efficiency and task completion ratio compared to the benchmark schemes.**

***Index Terms*—Cybertwin, deep reinforcement learning (RL), edge computing, federated learning (FL), resource allocation, task offloading.**

I. Introduction

RECENT advancements in Internet of Things (IoT) and wireless communication technology have enabled an explosive growth of smart terminal devices and data traffic. In the next 6G era, millions of end devices connect to the wireless network, accompanied by an increasing number of various applications (e.g., online games, virtual/augmented reality, and autonomous driving) [1]–[3]. These applications consume extensive computation resources and require extreme user Quality of Experience (QoE), low latency, and ultrareliable services [4], [5]. Furthermore, the enormous amount of data generated at the edge of wireless networks will converge into the core network, bringing unprecedented challenges to the existing wireless network architecture [6]. In this regard, the Cybertwin-based network architecture has recently proposed to be a potential solution in the 6G network that can support ultramassive connectivity and provide high-quality services for end devices.

Edge computing efficiently shifts computing and caching capabilities to the network edge for augmenting services and applications [7], [8]. Nevertheless, due to the limited resources in a single-edge cloud or end device, collaborative task execution of multiple edges or core clouds is needed. To accommodate the request, we can leverage Cybertwin to guide resource cooperation among end-edge-cloud and allocate computing, caching, as well as communication resources in a distributed way [9]. Specifically, Cybertwin serves as an intelligent agent located at the edge cloud that connects directly with end devices and provides high-quality service by offloading tasks to devices with higher computing power to meet end devices' diverse demands. Combined with the characteristics of Cybertwin, the related network architecture can support scalability, security, mobility, and availability.

However, the Cybertwin-based network architecture urgently needs to tackle some issues. The first problem is mixing task performance, since massive end devices generate various applications with diverse demands, and traditional hybrid offloading strategies lead to unbalanced consumption of wireless network resources and severe degradation of the Quality of Service (QoS). Second, multidimensional wireless resources are coupled and time varying [13]–[15]. The distributed and heterogeneous features of network resources cause the scheduling process to be increasingly complicated. Additionally, based on the current network state, network environment, and service requirements, more computing tasks are selectively accommodated and executed by end devices, edge computing servers, or cloud computing servers. Third, the interaction of heterogeneous end devices and migration across edge clouds can incur plenty of security and privacy problems [16]–[18]. End devices may not trust the edge and core clouds and even be reluctant to offload data related to computational tasks to servers. Meanwhile,

Manuscript received January 30, 2021; revised May 23, 2021; accepted July 3, 2021. Date of publication July 8, 2021; date of current version November 5, 2021. This work was supported in part by the National Key Research and Development Program under Grant 2018YFB0904900 and Grant 2018YFB0904905; in part by the Key Area Research and Development Program of Guangdong Province under Grant 2020B0101110003; and in part by the Shenzhen Science & Innovation Fund under Grant JCYJ20180507182451820. *(Corresponding authors: Hong Wen; Huanhuan Song.)*

Wenjing Hou, Hong Wen, Huanhuan Song, and Wenxin Lei are with the School of Aeronautics and Astronautics, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: uestc_hwj@126.com; sunlike@uestc.edu.cn; huanhuansong@126.com; leiwenxin@outlook.com).

Wei Zhang is with the School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia (e-mail: w.zhang@unsw.edu.au).

Digital Object Identifier 10.1109/JIOT.2021.3095677

complex and variable network connections and more intensive communication will increase the probability of information leakage.

To address the above thorny problems, a hierarchical task offloading and dynamic resource allocation scheme is proposed in this article for delay-tolerant and delay-sensitive missions by integrating edge computing and artificial intelligence (AI) into Cybertwin-based network. AI has been widely employed in wireless networks to solve high complexity optimization problems and make timely resource allocation decisions [19], [20]. Nonetheless, conventional AI algorithms require end devices to transmit their private data to edge servers or a remote cloud, which directly increases data and sensitive information leakage risk. Federated learning (FL) is regarded as a feasible approach for distributed machine learning to satisfy privacy protection and data security [21], [22]. It allows multiple participants to cooperatively build a shared model while retaining their privacy-sensitive data locally.

In this article, we integrate deep reinforce learning (DRL) and FL into Cybertwin-based network by jointly optimizing hierarchical task offloading and resource allocation and promoting the flexible collaboration of end devices, edge computing servers, or cloud computing servers for improving system processing efficiency and security. The main contributions in this article are summarized as follows.

1) *Hierarchical Task Offloading for Cybertwin-Based Network:* We propose the collaborative edge computing offloading and hybrid alternating offloading pattern for delay-sensitive tasks and delay-tolerant tasks, respectively.
2) *Various Offloading Pattern for Tasks and Typical Scenarios:* Different types of tasks can be processed by edge computing offloading and hybrid alternating offloading modes. For different offloading ways, we construct communication, computation, and cache models based on multidimensional network resources, system cost models based on time and energy, and efficiency models based on the hierarchical task offloading mechanism.
3) *Joint Optimization of Hierarchical Task Offloading and Resource Allocation Based on Multiagent Deep Deterministic Policy Gradient Algorithm (JHORA-MADDPG):* Joint optimization of the hierarchical task offloading and resource allocation problem is formulated to maximize the system processing efficiency. Since resource allocation and offloading task variables form high-dimensional matrices, traditional single-agent DRL methods are challenging to obtain the optimal policy with partial information and a dynamic network environment. In this article, a multiintelligent reinforcement learning (RL) algorithm is employed to solve this problem.
4) *FL Model for MADDPG:* A distributed model training approach based on the FL algorithm is investigated to solve the data security and privacy issues in the centralized training process of the multiagent deep deterministic policy gradient (MADDPG) model. This approach only shares the trained model parameters during the training process without the direct interaction of local data, protecting privacy while achieving the training effectiveness of traditional centralized training.

The remainder of this article is organized as follows. In Section II, we review the corresponding related works. The system model is given in Section III. Section IV presents communication, computation, cache, and system cost models. The problem formulation and MADDPG-based solution are provided in Section V. Numerical results are discussed in Section VI. Finally, Section VII concludes this work.

II. Related Works

A. Resource Allocation in Edge Computing

Resource allocation, including networking, caching, and computing, has received considerable attention. Yang *et al.* [23] studied the tradeoff between delay and energy consumption in the Maritime Internet of Things (M-IoT). A two-stage joint optimal offloading algorithm is proposed to allocate computation and communication resources. Ren *et al.* [24] investigated the collaborative cloud and edge computing scheme, where tasks can be partially processed by both the edge node and the cloud server. A general communication, computation, and caching framework was introduced in [25], where mobile devices form device-to-device (D2D) connections and share any combination of the three resources. Wu *et al.* [26] established a hybrid offloading model for edge and core cloud servers. Although these excellent works have investigated edge computing and resource allocation in wireless networks, they did not consider resource allocation for hierarchical task offloading with end-edge-cloud collaboration.

B. Artificial Intelligence-Enabled Edge Computing

Since AI and machine learning are emerging as powerful tools, some work has been done to explore how to design task offloading and resource allocation using AI in edge computing. Peng and Shen [27] formulated a joint vehicle association and multidimensional resource optimization problem in a vehicular network. They designed an RL solution to maximize the number of completed tasks. Li *et al.* [28] proposed a model-free DRL approach to reduce computational service latency and improve service reliability. He *et al.* [29] presented a novel DRL scheme for allocating network resources under the framework for trust-based social networks. Dai *et al.* [30] used the DRL-based computation offloading and resource allocation algorithm to minimize system energy consumption in a multiuser end-edge-cloud orchestrated network. Unfortunately, these works ignored end-user privacy protection and data security in the training process, while the long centralized training time may impact delay-sensitive applications.

C. Federated Learning-Enabled Edge Computing

Currently, some existing research on FL and edge computing aims to tackle the data security issue. Niknam *et al.* [31] discussed an FL framework for wireless communications. They adopted several use cases to introduce energy, bandwidth, delay, and data privacy concerns. In [32], DRL and FL

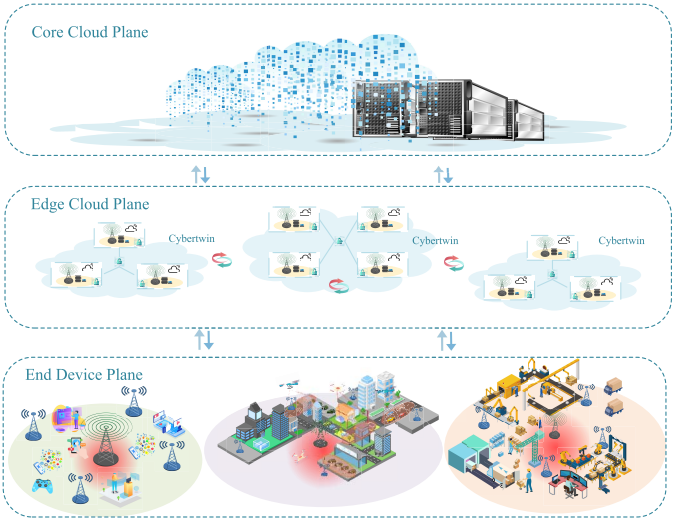

Fig. 1. Architecture of end-edge-cloud Cybertwin-based network.

were integrated into smart ocean FL IoT networks for resource allocation. Yu *et al.* [33] proposed a two-timescale deep RL method for joint optimization of computation offloading, resource allocation, and service caching placement with FL. However, incorporating FL to an end-edge-cloud Cybertwin-based network has not been widely studied.

In this article, we investigate a hierarchical task offloading and resource allocation strategy for end-edge-cloud collaboration in a Cybertwin-based network by leveraging FL to train the proposed MADDPG model in a distributed way for data security and privacy enhancement.

III. System Overview

A. Network Architecture

As shown in Fig. 1, we consider an end-edge-cloud Cybertwin-based network similar to [6] and [9], which consists of the end device plane, edge cloud plane, core cloud plane, and Cybertwin. In the end device plane, ends (smartphones, intelligent manufacturing equipment, UAVs, cars, etc.) generate various applications with diverse demands (e.g., online games, virtual/augmented reality, autonomous driving, and manufacturing). The computational tasks generated by the end device i at time slot t is given by $\mathcal{D}_i(t) = \{\varphi_i^{da}(t), \varphi_i^{co}(t), \varphi_i^{de}(t)\}$, where $\varphi_i^{da}(t)$, $\varphi_i^{co}(t)$, and $\varphi_i^{de}(t)$ denote the size of the task, the required computing resources, and the maximum tolerance delay, respectively. They can compute their tasks locally or offload tasks to anyone of their neighbor device via D2D communication.

In the edge cloud plane, multiple distributed edge servers are deployed on the edge of networks, responding to ends' requests faster and assisting core clouds in providing high-quality services. Each end device can offload tasks to its associated edge cloud or from the associated edge cloud to the neighboring edge cloud. The available spectrum, computation, and cache resources of the edge cloud $k \in \mathcal{K}$ are expressed as $\{C_k^{sp}, C_k^{co}, C_k^{ca}\}$. Note that per computing resource is defined as the CPU resource required to process one unit of computing task.

The core cloud plane consists of multiple core cloud servers. It is fully connected via high-speed optical links to constitute the core network that provides computing, caching, and communication resources to the end devices. Therefore, delay-tolerant and computing-intensive tasks can be offloaded on the central cloud servers to address the traffic and computing pressure on the edge cloud.

Cybertwin is a mirrored digital mapping of the end devices in virtual cyberspace, which serves as an intelligent agent located at the edge cloud that connects directly with end devices. It provides high-quality service and a more promising user experience by knowing requirements well on behalf of end devices and flexibly scheduling resources compared with the traditional edge computing architecture. For the end edge-cloud Cybertwin-based network, end devices first connect to the Cybertwin located in the edge cloud to acquire required services without the connection between ends and servers. Cybertwin provides efficient and high-quality services for end devices by managing various computing, caching, and communication resources in a distributed way from the cloud network (including the edge and core clouds) and offloading computing tasks to better computing devices. In short, Cybertwin acts as the assistant of end devices to achieve various required services.

In the above scenario, task execution owns five patterns: 1) local processing; 2) offloading to the edge cloud; 3) offloading to the nearby edge cloud; 4) offloading to the core cloud; and 5) offloading to the nearby device.

B. System Framework

Cybertwin can establish connections with end devices under its coverage to obtain real-time state information of end devices (e.g., task queues, positions, current computing, cache, and spectrum resources). It also collects the real-time state information of available resources on the edge cloud and the core cloud as a service agent. By providing the function of communications assistant, network behavior logger, and digital asset, Cybertwin high-efficiently supports online services. Thus, it serves as a scheduling agent to decide task offloading and resource allocation (e.g., spectrum, computing, and cache resource allocation) schemes as in Fig. 2.

Step 1: First, Cybertwin, located in the edge cloud, gathers real-time state information of task queue, end devices, edge cloud, and core cloud.

Step 2: According to the acquired state information, multidimensional resource management and system cost models are designed under the edge-dominated system for heterogeneous computing tasks. Furthermore, the problem of joint hierarchical task offloading and resource allocation is formulated for maximizing processing efficiency.

Step 3: A multiagent RL algorithm (MADDPG) is used to obtain the hierarchical task offloading and resource allocation scheme.

Step 4: Finally, Cybertwin transmits the corresponding offloading and resource allocation decisions to the end devices if computing tasks are processed locally or through other neighbor devices. Alternately, it will share the offloading

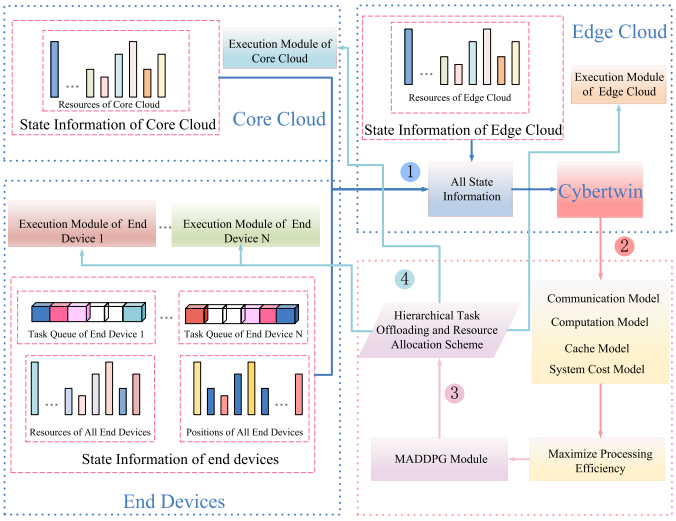

Fig. 2. System framework.

and allocation results to the edge cloud and core cloud if computing tasks are performed on the edge cloud and core cloud.

IV. System Model

Next-generation networks are expected to accommodate large-scale computing-intensive and latency-sensitive applications, which can be processed selectively on the end device, edge cloud, or core cloud. Most of the existing research concentrates on hybrid offloading strategies without differentiating delay-sensitive and delay-tolerant task, which inevitably generates excessive consumption of wireless network resources and severe degradation of QoS. Therefore, under the edge-dominated system, we design a hierarchical task offloading and multidimensional resource allocation strategy. For delay-sensitive tasks which cannot be processed by the cloud, the collaborative edge computing offloading mode, including local processing, offloading to the edge cloud, and offloading to the nearby edge cloud, is proposed. Additionally, we design hybrid alternating offloading for delay-tolerant tasks, including offloading to the cloud and offloading to the nearby device. Since the D2D offloading needs to match other devices with sufficient available resources, it is suitable for delay-tolerant tasks.

Let $\zeta_i \in \{0, 1\}$ denote the decision variable on the execution mode of task $\mathcal{D}_i$. If task $\mathcal{D}_i$ is processed by the collaborative edge computing offloading mode, $\zeta_i = 1$. If task $\mathcal{D}_i$ is processed by the hybrid alternating offloading mode, $\zeta_i = 0$. Furthermore, for delay-sensitive tasks, we define $\phi_{i,l}, \phi_{i,k}, \phi_{i,h,k} \in \{0, 1\}$ as the decision variable on the collaborative edge computing mode of task $\mathcal{D}_i$. If task $\mathcal{D}_i$ is processed locally, $\phi_{i,l} = 1$; otherwise, $\phi_{i,l} = 0$. If task $\mathcal{D}_i$ is offloaded to the edge cloud k, $\phi_{i,k} = 1$; otherwise, $\phi_{i,k} = 0$. If the edge cloud h offloads task $\mathcal{D}_i$ to edge cloud k, $\phi_{i,h,k} = 1$; otherwise, $\phi_{i,h,k} = 0$. For delay-tolerant tasks, we define $\gamma_{i,c}, \gamma_{i,j} \in \{0, 1\}$ as the decision variable on the hybrid alternating mode of task $\mathcal{D}_i$. If task $\mathcal{D}_i$ is offloaded to the core cloud, $\gamma_{i,c} = 1$; otherwise, $\gamma_{i,c} = 0$. If task $\mathcal{D}_i$ is offloaded to end device j by D2D communication, $\gamma_{i,j} = 1$; otherwise, $\gamma_{i,j} = 0$.

A. Communication Model

1) *End Device to Edge Cloud:* When the end device i selects to offload computing task $\mathcal{D}_i$ to the edge cloud $k \in \mathcal{K}$, the total available uplink spectrum resources are given by C_k^{sp}. The number of total uplink channels is denoted by N^k and the uplink channel set is represented by $\mathbb{N}^k$. Then, $c^k = C_k^{sp}/N^k$ indicates the bandwidth of each subchannel, and the available transmission rate of channel n assigned to end device i, can be expressed as

$$r_{i,n}^k(t) = c^k \log_2\left(1 + \frac{P_i^k g_{i,n}^k(t)}{\sigma^2 + \eta_{i,n}^k(t)}\right) \tag{1}$$

where P_i^k indicates the transmission power of end device i and edge cloud k. We employ $g_{i,n}^k(t)$ and $\eta_{i,n}^k(t)$ to represent the channel gain and interference of channel n, respectively. $x_{i,n}^k = 1$ represents channel n which is assigned to the end device; otherwise, $x_{i,n}^k = 0$. Then, the total uplink transmission rate between end device i and edge cloud k can be given as

$$r_i^k(t) = \sum_{n \in \mathbb{N}^k} x_{i,n}^k r_{i,n}^k(t). \tag{2}$$

2) *Interedge Cloud:* Due to the computing burden on the edge cloud, the hth end cloud cannot always own enough resources to meet the task requirement. Thus, the end cloud $h \in \mathbb{H}_k$ often offloads the task $\mathcal{D}_i$ to the nearby edge cloud $k \in \mathcal{K}$. Let $\mathbb{H}_k$ denote the set of neighboring edge clouds to end cloud k. Denote C_{k2k}^{sp} the total available uplink spectrum resources. N^{k2k} is the number of total uplink channels and $\mathbb{N}^{k2k}$ is the uplink channel set. Then, $c^{k2k} = C_{k2k}^{sp}/N^{k2k}$ indicates the bandwidth of each subchannel, and the available transmission rate of channel n assigned to end cloud k, can be described as

$$r_{i,n}^{h,k}(t) = c^{k2k} \log_2\left(1 + \frac{P_i^{h,k} g_{i,n}^{h,k}(t)}{\sigma^2 + \eta_{i,n}^{h,k}(t)}\right) \tag{3}$$

where $P_i^{h,k}$ indicates the transmission power of end cloud k and nearby edge cloud h. We employ $g_{i,n}^{h,k}(t)$ and $\eta_{i,n}^{h,k}(t)$ to represent the channel gain and interference of channel n, respectively. $x_{i,n}^{h,k} = 1$ denotes channel n which is assigned to the end cloud k for the communication between end cloud k and nearby edge cloud h; otherwise, $x_{i,n}^{h,k} = 0$. Then, the total uplink transmission rate between end cloud h and edge cloud k can be expressed as

$$r_i^{h,k}(t) = \sum_{n \in \mathbb{N}^{k2k}} x_{i,n}^{h,k} r_{i,n}^{h,k}(t). \tag{4}$$

3) *Interedge Device:* To fully exploit computation resources of end devices, the end device k can offload the task $\mathcal{D}_i$ to the nearby end devices with available computation resources and wireless resources by D2D communication. Denote $\mathbb{J}_i$ the set of neighboring edge devices

to end devices i. Let C_{d2d}^{sp} indicate the total available uplink spectrum resources. N^{d2d} is the number of total uplink channels and $\mathbb{N}^{d2d}$ is the uplink channel set. Then, $c^{d2d} = C_{d2d}^{sp}/N^{d2d}$ denotes the bandwidth of each subchannel, and the available transmission rate of channel n assigned to end device i, can be described as

$$r_{i,n}^{j}(t) = c^{d2d}\log_2\left(1 + \frac{P_i^j g_{i,n}^j(t)}{\sigma^2 + \eta_{i,n}^j(t)}\right) \tag{5}$$

where P_i^j indicates the transmission power of end device i and nearby edge device j. We employ $g_{i,n}^j(t)$ and $\eta_{i,n}^j(t)$ to represent the channel gain and interference of channel n, respectively. $x_{i,n}^j = 1$ represents channel n which is assigned to the end device j for the communication between end device i and nearby edge device j; otherwise, $x_{i,n}^j = 0$. Then, the total uplink transmission rate between end device i and edge device j can be expressed as

$$r_i^j(t) = \sum_{n\in\mathbb{N}^{d2d}} x_{i,n}^j r_{i,n}^j(t). \tag{6}$$

B. Computation Model

Collaborative Edge Computing Offloading Pattern: For end device i, the delay-sensitive tasks $\mathcal{D}_i$ can be executed in three ways, including local processing, offloading to the edge cloud, and offloading to the nearby edge cloud.

1) *Local Processing:* When task $\mathcal{D}_i$ selects to be executed locally, the total computing resource of end device i will be leveraged for task completion [34]. f_i is the computing resource of end device i. Thus, the local task delay can be described as

$$T_{i,l}^{\text{loc}}(t) = \frac{\varphi_i^{co}(t)}{f_i}. \tag{7}$$

$\kappa_i(f_i)^2$ indicates the energy consumption of unit computing resource on device i, and κ_i relays on the chip structure [34]. Subsequently, the local energy consumption of computing task $\mathcal{D}_i$ is given as follows:

$$E_{i,l}^{\text{loc}}(t) = \kappa_i\varphi_i^{co}(t)(f_i)^2. \tag{8}$$

2) *Offloading to the Edge Cloud:* Suppose that end devices under the same edge cloud coverage upload their missions to the edge cloud simultaneously. Each mission will be allocated for a portion of computing resources on the edge cloud. C_k^{co} is the total computing resource of end cloud k. λ_i^k is the proportion of the computing resources of the edge cloud k assigned to device i for processing task $\mathcal{D}_i$. Since the computing result has a short return time, we neglect the transmission time of the result [23]. Accordingly, the completion task delay includes transmission delay and execution delay, which can be depicted as

$$T_{i,k}^{\text{edge}}(t) = \frac{\varphi_i^{da}(t)}{r_i^k(t)} + \frac{\varphi_i^{co}(t)}{\lambda_i^k(t)C_k^{co}}. \tag{9}$$

Then, the energy consumption per computing resource for the edge cloud is denoted by e_k. The total energy consumption mainly consists of transmission and computing [35]. Thus, the total energy consumption of task $\mathcal{D}_i$ offloaded to edge cloud is expressed as

$$E_{i,k}^{\text{edge}}(t) = \frac{P_i^k\varphi_i^{da}(t)}{r_i^k(t)} + \varphi_i^{co}(t)e_k. \tag{10}$$

3) *Offloading to the Nearby Edge Cloud:* Due to the computational and storage limitation of the edge cloud h, the end cloud $h \in \mathbb{H}_k$ often offloads the ith end device's task to the nearby edge cloud k for higher computational resources. $\lambda_i^{h\to k}$ is the proportion of the computing resources of the edge cloud k assigned to device i from the edge cloud h. Similarly, the completion task delay includes transmission delay between the ith end device and the hth edge cloud, transmission delay between the hth edge cloud and the kth edge cloud, and execution delay can be expressed as

$$T_{i,h,k}^{\text{edge}}(t) = \frac{\varphi_i^{da}(t)}{r_i^h(t)} + \frac{\varphi_i^{da}(t)}{r_i^{h,k}(t)} + \frac{\varphi_i^{co}(t)}{\lambda_i^{h\to k}(t)C_k^{co}}. \tag{11}$$

Then, the total energy consumption of task $\mathcal{D}_i$ is depicted as

$$E_{i,h,k}^{\text{edge}}(t) = \frac{P_i^h\varphi_i^{da}(t)}{r_i^h(t)} + \frac{P_i^{h,k}\varphi_i^{da}(t)}{r_i^{h,k}(t)} + \varphi_i^{co}(t)e_k. \tag{12}$$

Hybrid Alternating Offloading Pattern: For end device i, the delay-tolerant tasks $\mathcal{D}_i$ can be executed in two ways, including offloading to the cloud and offloading to the nearby device.

1) *Offloading to the Core Cloud:* When end device i selects to offload its task to the core cloud directly, we ignore the execution time and energy consumption on the cloud due to a sufficient amount of computation and storage resources [23]. Thus, the completion task delay can be expressed as

$$T_{i,c}^{\text{cloud}}(t) = \frac{\varphi_i^{da}(t)}{r_i^k(t)} + \frac{\varphi_i^{da}(t)}{r_k^c(t)} \tag{13}$$

where $r_k^c(t)$ is the transmission rate between edge cloud k and the core cloud. P_k^c denotes the transmission power between edge cloud k and the core cloud. Then, the total energy consumption for task $\mathcal{D}_i$ is given as

$$E_{i,c}^{\text{cloud}}(t) = \frac{P_i^k\varphi_i^{da}(t)}{r_i^k(t)} + \frac{P_k^c\varphi_i^{da}(t)}{r_k^c(t)}. \tag{14}$$

2) *Offloading to the Nearby End Device:* When end device i selects to offload its task to the nearby end device $j \in \mathbb{J}_i$, similarly, the completion task delay includes transmission delay and matching processing delay between the ith end device and the jth edge device, execution delay, which can be described as

$$T_{i,j}^{d2d}(t) = \frac{\varphi_i^{da}(t)}{r_i^j(t)} + \frac{\varphi_i^{co}(t)}{f_j}. \tag{15}$$

Then, the energy consumption per computing resource for the end devices j is denoted by $\kappa_j(f_j)^2$, the total energy consumption of task $\mathcal{D}_i$ is depicted as

$$E_{i,j}^{d2d}(t) = \frac{P_i^j \varphi_i^{da}(t)}{r_i^j(t)} + \kappa_j \varphi_i^{co}(t)\left(f_j\right)^2. \tag{16}$$

C. Cache Model

If the end device i decides to offload task $\mathcal{D}_i$ to the edge cloud or the nearby edge cloud, a portion of the cache resources will be preallocated to the end device's task offloading request. The related data for each task should be cached before task execution. Let ω_i^k and $\omega_i^{h\to k}$ be the proportion of the cache resource assigned to the end device's task $\mathcal{D}_i$. Thus, the task is completed by the edge cloud or the nearby edge cloud, satisfying the following requirements:

$$\begin{aligned} C_k^{ca}\omega_i^k &\geq \varphi_i^{da}(t) \quad \forall k \in \mathcal{K} \\ C_h^{ca}\omega_i^{h\to k} &\geq \varphi_i^{da}(t) \quad \forall h \in \mathbb{H}_k. \end{aligned} \tag{17}$$

D. System Cost Model

For the end device i at time t, based on the above task execution for different offloading patterns, the total completion time of the task $\mathcal{D}_i$ can be expressed as

$$T_i(t) = \zeta_i(t) T_i^{\star}(t) + (1-\zeta_i(t)) T_i^{\diamond}(t) \tag{18}$$

where $T_i^{\star}(t)$ and $T_i^{\diamond}(t)$ denote the task completion time by the collaborative edge computing offloading and hybrid alternating offloading patterns, respectively. Then, the completion time $T_i^{\star}(t)$ is given as

$$\begin{aligned} T_i^{\star}(t) = \phi_{i,l} T_{i,l}^{\text{loc}}(t) &+ \sum_{k\in K} \phi_{i,k} T_{i,k}^{\text{edge}}(t) \\ &+ \sum_{h\in\mathbb{H}_k, k\neq h} \sum_{k\in K} \phi_{i,h,k} T_{i,h,k}^{\text{edge}}(t). \end{aligned} \tag{19}$$

Similarly, the completion time $T_i^{\diamond}(t)$ is denoted as

$$T_i^{\diamond}(t) = \gamma_{i,c} T_{i,c}^{\text{cloud}}(t) + \sum_{j\in\mathbb{J}_i, j\neq i} \gamma_{i,j} T_{i,j}^{d2d}(t). \tag{20}$$

The total energy consumption of the task $\mathcal{D}_i$ can be expressed as

$$E_i(t) = \zeta_i(t) E_i^{\star}(t) + (1-\zeta_i(t)) E_i^{\diamond}(t) \tag{21}$$

where $E_i^{\star}(t)$ and $E_i^{\diamond}(t)$ indicate the task energy consumption by the collaborative edge computing offloading and hybrid alternating offloading patterns, respectively. Then, the energy consumption $E_i^{\star}(t)$ is formulated as

$$\begin{aligned} E_i^{\star}(t) = \phi_{i,l} E_{i,l}^{\text{loc}}(t) &+ \sum_{k\in K} \phi_{i,k} E_{i,k}^{\text{edge}}(t) \\ &+ \sum_{h\in\mathbb{H}_k, k\neq h} \sum_{k\in K} \phi_{i,h,k} E_{i,h,k}^{\text{edge}}(t). \end{aligned} \tag{22}$$

Similarly, the energy consumption $E_i^{\diamond}(t)$ is denoted as

$$E_i^{\diamond}(t) = \gamma_{i,c} E_{i,c}^{\text{cloud}}(t) + \sum_{j\in\mathbb{J}_i, j\neq i} \gamma_{i,j} E_{i,j}^{d2d}(t). \tag{23}$$

The total system cost of computing tasks can be expressed as

$$Q_i(t) = \alpha_i E_i(t) + \beta_i T_i(t) \tag{24}$$

where α_i and β_i denote the weight coefficient of energy consumption and completion time, respectively. Mathematically, the relationship between the two weight coefficients is expressed as

$$\begin{aligned} \alpha_i + \beta_i &= 1 \\ 0 \leq \alpha_i &\leq 1 \\ 0 \leq \beta_i &\leq 1. \end{aligned} \tag{25}$$

V. Problem Formulation and Different Scenarios

A. Problem Formulation

Cybertwin acts as a communication assistant, network behavior logger, and digital asset owner for the entire network architecture. Due to the limited resources of edge clouds or end devices under traditional edge computing, it is usually required to execute tasks by multiple edge clouds or core clouds. Therefore, we leverage Cybertwin to guide resource collaboration among end-edge-cloud and formulate task offloading and resource allocation problems. To measure the edge-dominated Cybertwin network's performance, we consider two factors, i.e., the system cost as well as the number of completed tasks to formulate the processing efficiency of the system, which can be depicted as

$$\eta(t) = \sum_{i\in\mathcal{N}(t)} \frac{|M_i(t)|}{Q_i(t)} \tag{26}$$

where $\mathcal{N}(t)$ and $M_i(t)$ denote the number of end devices and completed tasks for the system at time slot t, respectively. We aim to decrease the system cost while increasing the number of completed tasks. The objective of the optimization problem is to maximize the system's processing efficiency with the constraints of the task requirement, hierarchical computing offloading, and multidimensional resource. The combinatorial optimization problem is represented as

$$\max_{\zeta(t),\phi(t),\gamma(t)\lambda(t),\omega(t)} \quad \eta(t) \tag{27}$$

$$\text{s.t.} \quad \lambda_i^k(t), \omega_i^k(t) \in [0,1] \tag{27a}$$

$$\zeta_i(t), \phi_{i,l}(t), \phi_{i,k}(t), \phi_{i,h,k}(t), \gamma_{i,c}(t) \gamma_{i,j}(t) \in \{0,1\} \tag{27b}$$

$$\zeta_i\left(\phi_{i,l}(t) + \sum_{k\in K}\phi_{i,k}(t) + \sum_{h\in\mathbb{H}_k, k\neq h}\sum_{k\in K}\phi_{i,h,k}(t)\right) \leq 1 \tag{27c}$$

$$(1-\zeta_i)\left(\sum_{j\in\mathbb{J}_i, i\neq j}\gamma_{i,j} + \gamma_{i,c}\right) \leq 1 \tag{27d}$$

$$\lambda_i^k(t) + \sum_{h\in\mathbb{H}_k} \lambda_i^{h\to k}(t) \leq 1 \quad \forall k \in K \tag{27e}$$

$$\omega_i^k(t) + \sum_{h\in\mathbb{H}_k} \omega_i^{h\to k}(t) \le 1 \quad \forall k \in K \quad (27f)$$

$$\sum_I x_{i,n}^j \le 1 \quad \forall i \in I, j \in \mathbb{J}_i, n \in \mathbb{N}^{d2d} \quad (27g)$$

$$\sum_I x_{i,n}^k \le 1 \quad \forall i \in I, k \in K, n \in \mathbb{N}^k \quad (27h)$$

$$\sum_K x_{i,n}^{h,k} \le 1 \quad \forall k \in K, h \in \mathbb{H}_k, n \in \mathbb{N}^{k2k} \quad (27i)$$

$$T_i(t) \le \varphi_i^{de}(t) \quad \forall i \in I. \quad (27j)$$

Constraint (27a) represents that the allocation variables of computing and cache resources for the edge cloud. Constraint (27b) denotes the offloading decisions of two computing patterns, including five modes. Constraints (27c) and (27d) indicate end devices only choose one way to handle their tasks. Constraints (27e) and (27f) represent that the allocation of computing and cache resources on the edge cloud cannot exceed the total resources. Constraints (27g), (27h), and (27j) imply each channel should be allocated to only one end device or edge cloud for the end device to edge cloud, interedge cloud, and D2D communication at each time slot. Constraint (27k) denotes the task delay need to satisfy the task's delay requirement.

B. Different Scenarios

In the Cybertwin-based network architecture, we design hierarchical task offloading with end-edge-cloud collaboration for delay-sensitive tasks and delay-tolerant tasks. In addition, we discuss four task execution cases in terms of the collaborative edge computing offloading and hybrid alternating offloading modes and further analyze the four computing scenarios. First, we consider the four cases of task execution processing in the system as follows.

1) *Case 1:* Both collaborative edge computing offloading side and hybrid alternating offloading side simultaneously have computing tasks being processed.
2) *Case 2:* The collaborative edge computing offloading side has computing tasks being processed, while the hybrid alternating offloading side does not have computing tasks.
3) *Case 3:* The hybrid alternating offloading side has computing tasks being processed, while the collaborative edge computing offloading side does not have computing tasks.
4) *Case 4:* There are no computing tasks on both the hybrid alternating offloading and the collaborative edge computing offloading side.

Next, we assume $t = 0$. There are tasks to be processed on both the hybrid alternating offloading side and the edge cooperative offloading side (i.e., case 1). Thus, when $t \in [0, T]$, four computing scenarios exist in the system, as shown in Fig. 3.

1) *Scenario 1:* At time $t \in [0, T]$, all computing tasks on the collaborative edge computing offloading side have been completed, while the hybrid alternating computing side has computing tasks to be addressed (i.e., case 3). After the time t, the remaining computing tasks on the hybrid alternating offloading side are handed over to the edge computing and hybrid alternating offloading side for cooperative execution (i.e., case 1).
2) *Scenario 2:* At time $t \in [0, T]$, all computing tasks on the collaborative edge computing offloading and hybrid alternating computing sides have been completed (i.e., case 4). Thus, the edge computing and hybrid alternating offloading side wait for the computing task to come in the next time slot $T + 1$.
3) *Scenario 3:* At time $t = T$, the computing tasks have not been completed on both the collaborative edge computing offloading and hybrid alternating offloading sides (i.e., case 1). The remaining computing tasks will then be added to the task queue of the collaborative edge computing offloading and hybrid alternating offloading sides in the next time slot $T + 1$, respectively (the remaining tasks are placed at the head of the queue).
4) *Scenario 4:* At time $t = T$, the computing tasks have not been completed on the collaborative edge computing offloading, while hybrid alternating offloading sides' tasks have been completed (i.e., case 2). Subsequently, the remaining computing tasks will be added to the task queue of the collaborative edge computing offloading in the next time slot $T+1$ (the remaining computing tasks are placed at the head of the queue).

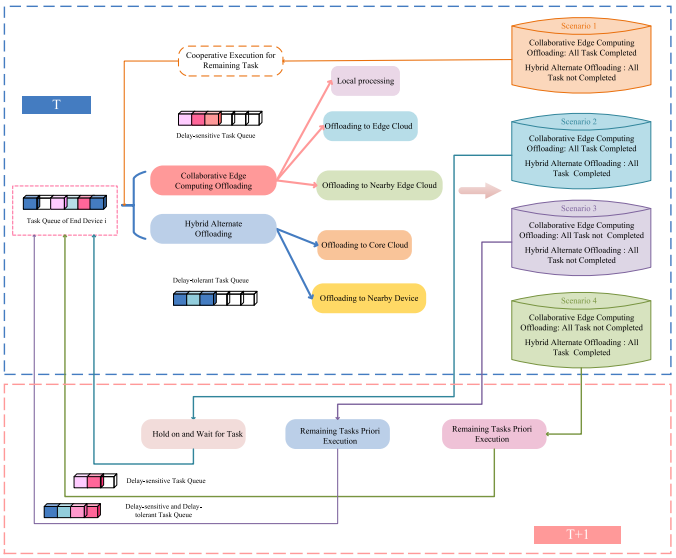

Fig. 3. Different scenarios of hierarchical task offloading.

VI. MADDPG for the JHORA Problem

A. MADDPG-Based Solution

The above optimization problem intends to complete as many tasks as possible while reducing the system processing cost. From (27), the assignment of communication, cache, and computational resources is coupled with each other. An increasing number of end devices directly leads to the higher computational complexity of the problem. Traditional

optimization methods are hard to solve this optimization problem rapidly. Therefore, we employ an RL approach to cope with the proposed task hierarchical offloading and resource allocation problem.

In RL, the agent learns how to interact with the environment, automatically optimize its policy based on the observation of rewards. As a model-free RL algorithm, Q-learning has been widely exploited to solve various complex sequential decision problems in wireless networks. However, RL takes much time to obtain the optimal policy since it needs to know the whole state of the environment, and RL is not suitable for networks with large-scale end devices access. Integrating deep learning and RL is regarded as a pivotal solution to solve the above problem. Deep RL trains learning models by utilizing deep neural networks (DNNs) for faster learning speed and better performance.

Based on the main idea of deep RL, the above optimization problem is transformed into a Markov decision process (MDP) to represent the hierarchical task offloading and multidimensional resource allocation decision-making process. Nevertheless, due to heterogeneous multidimensional resources and time-varying network environments, the traditional single-agent approach cannot learn collaborative hierarchical offloading and resource allocation strategies for end-edge-cloud to satisfy various end devices' requirements.

We design an MADDPG-based solution in which Cybertwin, located on the edge cloud, acts as an agent to learn collaborative hierarchical offloading and resource allocation schemes. In the policy update process, each Cybertwin agent should consider the actions of other Cybertwin agents instead of only using its action to update the policy. We train the model by the MADDPG method. Then, we gain the optimal offloading and resource allocation policy according to the trained model. In the following, we introduce the MDP, including the state space, action space, and reward function.

1) Observation State: At the beginning of each time slot, Cybertwin collects the computing task request from the end device under its coverage and currently available resources. Then, Cybertwin establishes virtual task queues. For the edge cloud k, we define the observations state which consists of the following elements at time slot t.

1) Let $O_k^{\text{task}}(t) = (\mathcal{D}_{1,k}(t), \mathcal{D}_{2,k}(t), \ldots, \mathcal{D}_{N(t),k}(t))$ denote the state of unprocessed tasks managed by Cybertwin, located on the edge cloud k. At time t, if end device i is far away from the coverage of the edge cloud k, we define $\mathcal{D}_{i,k}(t) = 0$.
2) At time t, the available spectrum resources managed by Cybertwin, located on the edge cloud k, are represented as $O_k^{sp}(t) = C_k^{sp}(t)$.
3) At time t, the available computing resources managed by Cybertwin, located on the edge cloud k, are represented as $O_k^{co}(t) = C_k^{co}(t)$.
4) At time t, the available cache resources managed by Cybertwin, located on the edge cloud k, are represented as $O_k^{ca}(t) = C_k^{ca}(t)$.

The observation state of Cybertwin, located on the edge cloud k, at time t, can be described as $O_k(t) = (O_k^{\text{task}}(t), O_k^{sp}(t), O_k^{co}(t), O_k^{ca}(t))$. Thus, a set of system observation can be defined as

$$O(t) = (O_1(t), \ldots, O_K(t)). \tag{28}$$

In this problem, the environment is fully observed, thus the observations are equivalent to the environment state.

2) Action Space: At time t, the action $a(t)$ contains the collaborative edge computing offloading and hybrid alternating offloading strategy selected by Cybertwin for the end device i. Cybertwin makes decisions on whether to execute tasks locally, offload tasks to the edge cloud and nearby edge cloud, offload tasks to the core cloud and other end devices via obtaining multidimensional resources from end device, edge clouds, and core cloud. For Cybertwin, at the beginning of each time slot, collecting tasks under its coverage is denoted as $O_k^{task}(t) = \sum_{i \to k} \mathcal{D}_i(t)$. For end device $\mathcal{N} = \{1, 2, \ldots, N\}$, it only can select one way to handle its task $\mathcal{D}_i(t)$ at each time slot. The decision variable on the execution mode is $\phi_{i,l}(t), \phi_{i,k}(t), \phi_{i,h,k}(t), \gamma_{i,c}(t), \gamma_{i,j}(t) \in \{0, 1\}$. Therefore, we can define the constraint $\phi_{i,l}(t) + \phi_{i,k}(t) + \phi_{i,h,k}(t) + \gamma_{i,c}(t) + \gamma_{i,j}(t) \in \{0, 1\}$. Then, we analyze the transform processing of five processing modes as follows.

1) When task $\mathcal{D}_i(t)$ is executed on the edge cloud k, we define $\phi_{i,k}(t) = 1$; otherwise, $\phi_{i,k}(t) = 0$. Let $\lambda_{i,k}(t)$ and $\omega_{i,k}(t)$ be the allocated computing and cache resources for task $\mathcal{D}_i(t)$.
2) When task $\mathcal{D}_i(t)$ is allocated to the edge cloud k from the edge cloud h, we define $\phi_{i,h,k}(t) = 1$; otherwise, $\phi_{i,h,k}(t) = 0$.
3) When $\mathcal{D}_{i,k}(t) \neq 0$, Cybertwin, located on the edge cloud k manages the task request of end device i. If task $\mathcal{D}_i(t)$ is assigned to locally execution, i.e., $\phi_{i,l}(t) = 1$. Thus, we define $\phi_{i,l,k}(t) = 1$ as locally execution; otherwise, $\phi_{i,l,k}(t) = 0$.
4) When $\mathcal{D}_{i,k}(t) \neq 0$, task $\mathcal{D}_i(t)$ is assigned to the core cloud for execution, i.e., $\gamma_{i,c}(t) = 1$. Thus, we define $\gamma_{i,c,k}(t) = 1$ as the core cloud for execution; otherwise, $\gamma_{i,c,k}(t) = 0$.
5) When $\mathcal{D}_{i,k}(t) \neq 0$, task $\mathcal{D}_i(t)$ is assigned to other end devices for execution, i.e., $\gamma_{i,j}(t) = 1$. Thus, we define $\gamma_{i,j,k}(t) = 1$ as D2D execution; otherwise, $\gamma_{i,j,k}(t) = 0$.

In short, we can obtain the constraint $\phi_{i,l,k}(t) + \phi_{i,k}(t) + \phi_{i,h,k}(t) + \gamma_{i,c,k}(t) + \gamma_{i,j,k}(t) \in \{0, 1\}$. For Cybertwin, we define its action as $a_k(t) = (\phi_{i,l,k}(t), \phi_{i,k}(t), \phi_{i,h,k}(t), \gamma_{i,c,k}(t), \gamma_{i,j,k}(t), \lambda_{i,k}(t), \omega_{i,k}(t))$, the system action is expressed as

$$a(t) = (a_1(t), \ldots, a_K(t)). \tag{29}$$

3) Reward Function: To enhance the system's processing efficiency for the tradeoff between the number of completed tasks and the cost of system, we consider the processing efficiency in the Cybertwin-based framework as the system reward.

Cybertwin of each edge cloud k is an agent. For each edge cloud k, the processing efficiency is represented as $\eta_k(t) = [(\sum_{i:\mathcal{D}_{i,k}(t)\neq 0} |M_i(t)|)/(\sum_{i:\mathcal{D}_{i,k}(t)\neq 0} Q_i(t))]$, associated with the cost of time and resource (i.e., spectrum, computing, and cache resource) for task execution, and the number of completed tasks. Thus, according to the constraints and

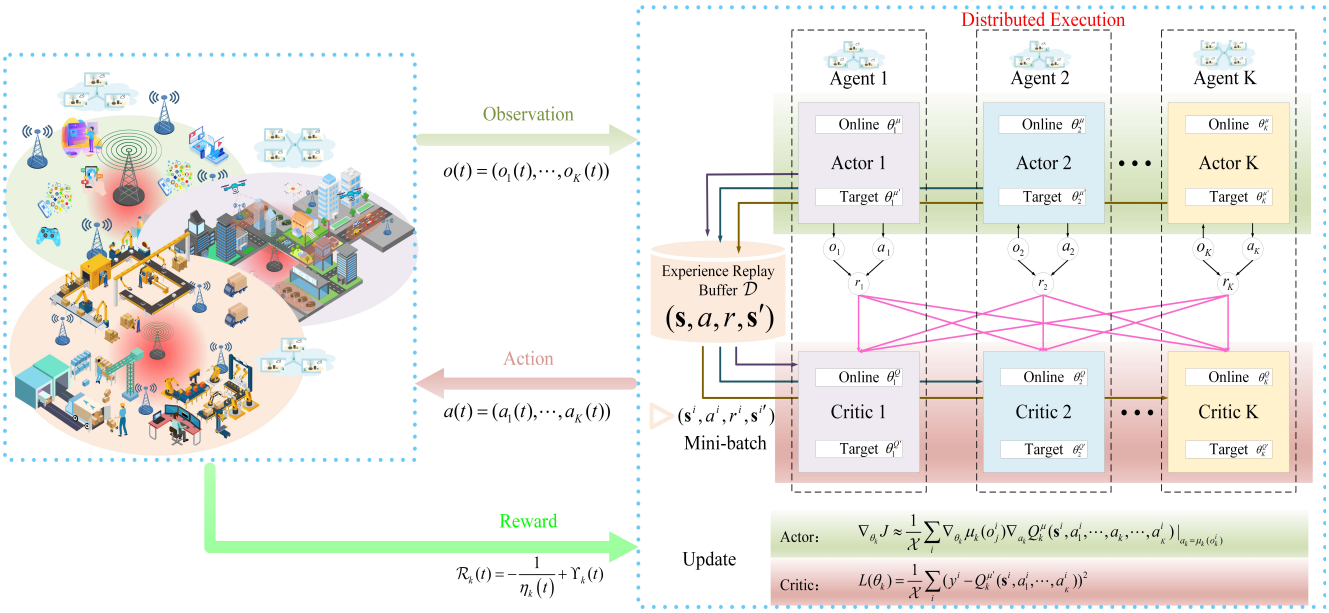

Fig. 4. MADDPG framework for JHORA in cybertwin-based networks.

objective functions, we denote the reward as

$$\mathcal{R}_k(t)=-\frac{1}{\eta_k(t)}+\Upsilon_k(t). \tag{30}$$

Note that $\Upsilon_k(t)$ represents the failure penalty for each Cybertwin of edge cloud k, which is related to the four scenarios of hierarchical task processing schemes, and the failure penalty emerges if assigned tasks are not completed within the time slot.

B. Joint Hierarchical Offloading and Resource Allocation Algorithm Based on MADDPG

We present the JHORA-MADDPG framework in this section, as shown in Fig. 4. MADDPG is an algorithm based on actor–critic. Each Cybertwin agent K owns the actor and critic with two DNNs, namely, the target network and the evaluation network, in order to obtain the hierarchical offloading and resource allocation strategy by the deep deterministic policy gradient (DDPG) method. The actor makes action decisions on the basis of the agent's state, and the critic is responsible for evaluating the actor's behavior. Thus, the target and evaluation networks are used to update parameters θ_K^μ and θ_K^Q. Cybertwin's experience replay buffer $\mathcal{D}$ is leveraged to save transitions related to observations and action for the training stage. When updating the parameters of target and evaluation networks, the evaluation network can randomly obtain the transitions from Cybertwin's experience replay buffer. The experience replay and actor–critic network can improve the stability of the DDPG training process while breaking the correlation of training data.

DDPG aims to obtain optimal policies and learns the corresponding action functions through interaction with the environment. The pseudocode of the proposed MADDPG algorithm for the JHORA problem is illustrated in Algorithm 1. During the training state, assume that $\theta=\{\theta_1,\ldots,\theta_k\}$ is the parameter set of networks for the K agents. The parameter set of the corresponding deterministic policy is represented by $\mu=\{\mu_{\theta_1},\ldots,\mu_{\theta_k}\}$. Thus, the gradient of the deterministic policy for agent k can be represented as

$$\nabla_{\theta_k}J(\mu_k)=\mathbb{E}_{s,a\sim\mathcal{D}}\left[\nabla_{\theta_k}\mu_k(o_j)\nabla_{a_k}Q_k^\mu(\mathbf{s},a_1,\cdots,a_K)\big|_{a_k=\mu_k(o_k)}\right] \tag{31}$$

where $\mathcal{D}$ is the experience replay with $(\mathbf{s},a,r,\mathbf{s}')$. The Q-function is denoted as $Q_k^\mu(\mathbf{s},a_1,\ldots,a_k,\ldots,a_K)$. In addition, the critic updates the loss function of the target Q-function as follows:

$$\begin{aligned}L(\theta_k)&=\mathbb{E}_{\mathbf{s},a,r,\mathbf{s}'}\left(y-Q_k^{\mu'}(\mathbf{s},a_1,\ldots,a_K)\right)^2\\ y&=r_k+\gamma Q_k^{\mu'}(\mathbf{s},a_1',\ldots,a_K')\big|_{a_j'=\mu_j'(o_j')}\end{aligned} \tag{32}$$

where γ is denoted as the discount factor. The action network is updated by minimizing the policy gradient of the agent, which can be expressed as

$$\nabla_{\theta_k}J\approx\frac{1}{\mathcal{X}}\sum_i\nabla_{\theta_k}\mu_k(o_j^i)\nabla_{a_k}Q_k^\mu(\mathbf{s}^i,a_1^i,\ldots,a_K^i)|_{a_k=\mu_k(o_k')} \tag{33}$$

where i denotes the index of samples, and $\mathcal{X}$ represents the size of mini-batch. Finally, the target network parameters can be updated as

$$\theta_k=\kappa\theta_k+(1-\kappa\theta_k'). \tag{34}$$

For Algorithm 1, one can observe that the training algorithm mainly consists of the actor network, critic network, and replay buffer. The actor and critic networks of each Cybertwin agent K are performed via two DNNs, namely, the target network and the evaluation network. Thus, the complexity of

Algorithm 1: MADDPG Algorithm for JHORA

Initialize: the actor's evaluation network and critic's target network for the agent;
for *episode = 1 : M* **do**
  Initialize a random process $\mathcal{N}$ for exploration of action;
  Receive initial observations o;
  Set initial state $\mathbf{s}$ and $r^{\star}$=0;
  for *t = 1 : max-episode-length* **do**
    Each agent k chooses action $a_k(t) = \mu_{\theta_k}(o_k(t)) + \mathcal{N}_t$ by the current policy and exploration;
    Execute actions $a(t) = (a_1(t), \cdots, a_K(t))$;
    Obtain reward $r(t)$ and new observation $o'_k(t)$;
    Input the new state $s'_k(t)$ to each agent k;
    Store $(\mathbf{s}, a(t), r(t), \mathbf{s}')$ into the reply buffer $\mathcal{D}$;
    $\mathbf{s} \leftarrow \mathbf{s}'$
    for *agent* $k = 1 : K$ **do**
      Sample a random minibatch of $\mathcal{X}$ samples $\left(\mathbf{s}^i, a^i, r^i, \mathbf{s}^{i'}\right)$ from $\mathcal{D}$;
      Set $y^i = r^i_k + \gamma Q^{\mu'}_k\left(\mathbf{s}^i, a'_1, \cdots, a'_K\right)\Big|_{a'_j=\mu'_j\left(o^i_j\right)}$;
      Update critic's evaluation network through minimizing the loss:
      $L(\theta_k) = \frac{1}{\mathcal{X}}\sum_i\left(y^i - Q^{\mu'}_k\left(\mathbf{s}^i, a^i_1, \cdots, a^i_K\right)\right)^2$;
      Update actor's evaluation network utilizing the sampled policy gradient based on Eq.(33)
    end
    Update target network parameters for each agent: $\theta_k \leftarrow \kappa\theta_k + \left(1 - \kappa\theta'_k\right)$;
  end
  $r^{\star} = r^{\star} + r^{\star}(t)$
end

Algorithm 1 is mainly affected by four neural networks, which can be expressed as [36]

$$2 \times \sum_{j=0}^{J} n_{\text{actor},j} \cdot n_{\text{actor},j+1} + 2 \times \sum_{l=0}^{L} n_{\text{critic},l} \cdot n_{\text{critic},l+1} = O\left(\sum_{j=0}^{J} n_{\text{actor},j} \cdot n_{\text{actor},j+1} + \sum_{l=0}^{L} n_{\text{critic},\,l} \cdot n_{\text{critic},\,l+1}\right) \quad (35)$$

where J and L are assumed to be the number of fully connected layers for the actor DNN network and critic DNN network, respectively. $n_{\text{actor},j}$ and $n_{\text{critic},\,l}$ denote the unit number in the jth actor layer and the critic lth layer. The input size is $n_{\text{actor},0}$ and $n_{\text{critic},\,0}$.

C. Distributed MADDPG Model Based on Federated Learning

The proposed MADDPG-enabled joint hierarchical offloading and resource allocation algorithm aims to make it possible to process as many tasks as possible while significantly reducing the system processing cost. Under the proposed scenario, end devices send their task offloading requests to the agent on the edge of the network. Thus, a large amount of end device privacy information is aggregated in Cybertwin. However, when training a multiagent deep RL model, end devices may not trust Cybertwin and provide detailed task offloading requests (preserving privacy-sensitive data locally) to Cybertwin. It brings a significant challenge for hierarchical offloading and resource allocation.

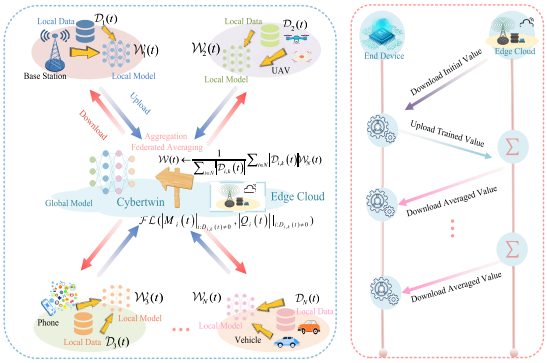

Fig. 5. FL model in Cybertwin-based networks.

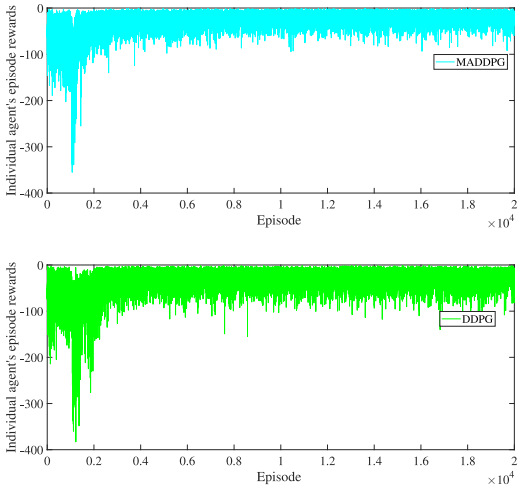

Fig. 6. Rewards achieved per episode of individual agent.

To solve the sensitive information leakage issue and relieve the computational pressure at the edge while improving the edge computing-oriented Cybertwin network's performance, we construct an FL model in Cybertwin-based networks, as shown in Fig. 5. In the initial phase, end devices obtain the global MADDPG model $\mathcal{W}(t)$ from the associated Cybertwin, then end devices train the local model $\mathcal{W}_1(t), \mathcal{W}_2(t), \ldots, \mathcal{W}_N(t)$ based on local data and global model. Next, local model parameters will be transmitted to the Cybertwin, then Cybertwin aggregates the updated parameters and conducts federated averaging $\mathcal{FL}(|M_i(t)|_{i:D_{i,k}(t)\neq 0}, |Q_i(t)||_{i:D_{i,k}(t)\neq 0})$ for an updated global MADDPG model $\mathcal{W}(t+1)$. The pseudocode of the proposed FL-based MADDPG algorithm is presented in Algorithm 2.

VII. Numerical Results

A. Simulation Setting

In this section, Python and TensorFlow are utilized to implement an MADDPG-based algorithm for hierarchical

Algorithm 2: FL-Based MADDPG Training Model for JHORA

Initialize:
Cybertwin side:
At the beginning of the learning phase $t = 0$, initialize the weight values of the MADDPG model $\mathcal{W}(0)$;
End device side:
Initialize the weight values of the local MADDPG model $\mathcal{W}_i(0)$, $(i = 1, 2, \ldots, N)$;
Obtain $\mathcal{W}(0)$ from the Cybertwin;
Denote $\mathcal{W}_i(0) = \mathcal{W}(0)$,$(i = 1, 2, \cdots, N)$;
for $t = 1$: *max-episode-length* **do**
  function $\mathcal{FL}\Big(|M_i(t)|_{i:D_{i,k}(t)\neq 0}, |Q_i(t)||_{i:D_{i,k}(t)\neq 0}\Big)$;
  End device side:
  for $i = 1 : N$ **do**
    Obtain $\mathcal{W}(t)$ from the Cybertwin;
    Denote $\mathcal{W}_i(t) = \mathcal{W}(t)$;
    Execute locally training based on $\mathcal{W}_i(t)$ with the current value $\mathcal{FL}\Big(|M_i(t)|_{i:D_{i,k}(t)\neq 0}, |Q_i(t)||_{i:D_{i,k}(t)\neq 0}\Big)$;
    Update the trained weight values $\mathcal{W}_i(t+1)$ and transmit corresponding values to Cybertwin;
  end
  Cybertwin side:
  for $k = 1 : K$ **do**
    Aggregate the updated weight values $\mathcal{W}_i(t)$ of end devices under its coverage;
    Conduct federated averaging (FedAvg);
    Broadcast the averaged weight value $\mathcal{W}_i(t+1)$;
  end
  end function;
end

TABLE I
TRAINING PARAMETERS

Parameter	Value
Layer type of critic	Fully connected
Hidden layers of critic networks	4
Neurons of hidden layers for critic networks	[2048,1024,512,256]
Layer type of actor	Fully connected
Hidden layers of actor networks	2
Neurons of hidden layers for actor networks	[512,256]
Activation function	Adam
Optimization method	Tanh, Relu
Learning Rate of critic	0.0001
Learning Rate of actor	0.0001
The size of mini-batch	64
The size of replay buffer	20000
Discount factor	0.95

task offloading and resource management. Compared to benchmark algorithms, we verify the effectiveness of the proposed JHORA-MADDPG scheme in several simulations. Specifically, Considering a Cybertwin-based network with one core cloud, $k = 3$ edge clouds, and $N = 100$–300 end devices randomly distributed under the edge cloud's coverage. Each edge cloud is composed of four Intel next units of computing (NUC) as agents connected via wireless networks. Then, we set the time slot interval to 2 s. In the JHORA-MADDPG scheme, the evaluation network consists of one input layer,

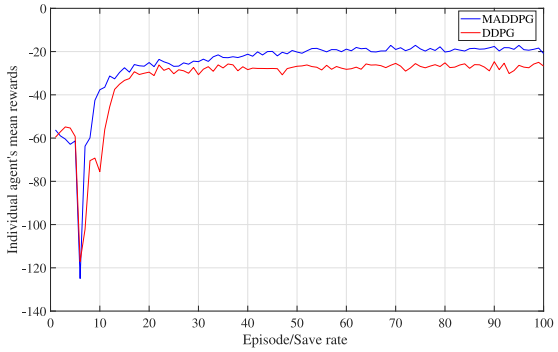

Fig. 7. Average rewards of different algorithms.

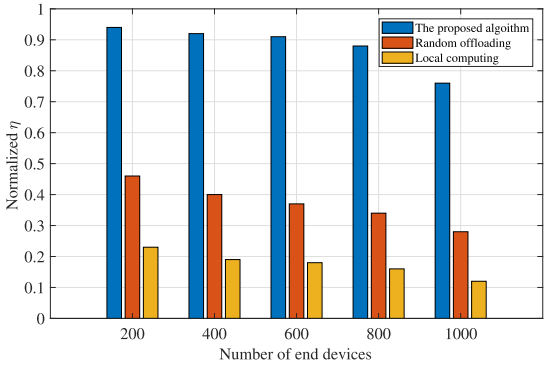

Fig. 8. System processing efficiency of different algorithms.

four fully connected hidden layers, and one output layer. Furthermore, in the output layer, each agent's actor adopts the tanh function to constrain the output value. We train the MADDPG model based on Algorithm 1 and set the maximum number of episodes as 20 000. The related training parameters of the neural network are presented in Table I.

To evaluate the proposed algorithm's performance, we compare it with the following benchmark algorithms as follows.

1) *Deep Deterministic Policy Gradient [37]:* Assume that one agent centrally manages the state information of all end devices, their corresponding edge clouds, as well as the core cloud in the system. The single agent's action is the offloading decision and resource allocation strategy for all the end devices.
2) *Random Task Offloading (Random)*: Latency-tolerant and latency-sensitive tasks are allocated randomly to the local execution, edge cloud execution, offloading from the edge cloud to the nearby edge cloud execution, D2D execution, and core cloud execution.
3) *Local Computing (LP)*: All computation tasks are processed locally.

B. Performance Analysis

We train different Cybertwin agents based on the MADDPG algorithm. Fig. 6 presents the rewards achieved per episode for the individual agent. Obviously, the rewards of Cybertwin increase with the number of training episodes. In the 0–2000 episodes, the rewards of Cybertwin are small and drastically fluctuate, after which both the MADDPG and DDPG algorithms can achieve the convergence state. Compared with the

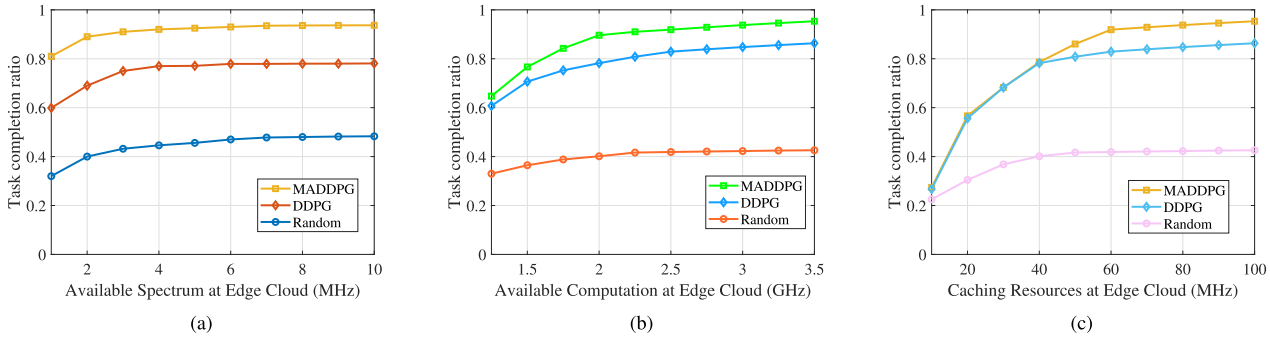

Fig. 9. Task completion ratio versus the amounts of spectrum, computational, and caching resources under different algorithms. (a) Task completion ratio versus spectrum resources. (b) Task completion ratio versus computing resources. (c) Task completion ratio versus caching resources.

DDPG algorithm, the rewards of Cybertwin by the MADDPG algorithm fluctuate in a smaller range and converge faster.

The individual agent's average rewards under the MADDPG and DDPG algorithms are shown in Fig. 7. Specifically, the number of episodes is 20 000, and the save rate is set to 200. With the increase in the training episodes/save rate, the average reward of the MADDPG algorithm always performs higher than the DDPG algorithm, eventually obtains a stable reward. The reason is that the proposed MADDPG algorithm owns different target and evaluation networks for each Cybertwin that can leverage the characteristics of Cybertwin and implement a better hierarchical task offloading and resource allocation strategy. Note that the DDPG algorithm has only one target and evaluation network for all Cybertwins.

Fig. 8 shows the system processing efficiency of different algorithms for the different number of end devices. With the increase in the number of end devices, the system processing efficiency of all strategies decreases. This is because the increment of end devices will generate more computation tasks with diverse requirements to compete for the limited spectrum, computation, and cache resources of the edge cloud, which results in some computation tasks cannot be processed before the deadline. Due to the fixed amounts of spectrum, computation, and cache resources in the edge cloud, more latency-sensitive tasks will be executed locally. Most of the latency-tolerant tasks can only be allocated to hybrid alternating strategies for execution rather than being selected for collaborative edge computing offloading strategies. Thus, system processing cost increases, and the system processing efficiency gradually decreases accordingly.

Meanwhile, it can be seen that the proposed JHORA-MADDPG algorithm can ensure that the system processing efficiency is always higher than the Random and Local computing algorithm. As the number of end devices increases, the system processing efficiency of the MADDPG algorithm decreases more slowly, compared with the Random and DDPG algorithms. The reason is that the proposed JHORA-MADDPG algorithm can efficiently utilize the resources of end devices, edge clouds, and core clouds according to the requirements of different computing tasks such that the system can process as many tasks as possible.

In Fig. 9(a), it can be seen that the task completion ratio of the proposed JHORA-MADDPG algorithm, DDPG, and Random algorithms slowly increases as the spectrum resource of the edge cloud increases from 1 to 10 MHz and eventually achieves a stable task completion ratio. In addition, the task completion ratio of the proposed JHORA-MADDPG algorithm with edge computing dominant scheme is higher than those of the DDPG and random algorithms. This is because computing tasks are completed with the satisfied latency requirement, determined by the spectrum and computational resource. More spectrum resources are leveraged to satisfy computing tasks' latency requirements as utilizing more computational resources, which is beneficial to the collaborative edge computing offloading strategy.

In Fig. 9(b), the results show that the task completion ratio of different algorithms with the edge cloud's different computing resources. As the edge cloud's computational resources increase, the increment of computational resources available for the collaborative edge computing offloading strategy can satisfy more tasks. Also, the proposed MADDPG algorithm performs better than Random and DDPG algorithms in terms of task completion ratio, and this is because the proposed JHORA-MADDPG algorithm enables hierarchically offload tasks with different requirements under the dominance of edge computing and more accurately allocate computational resources. For the MADDPG-based algorithm, different Cybertwin agents are trained cooperatively to achieve the maximum reward.

In Fig. 9(c), the task completion ratio of the proposed JHORA-MADDPG algorithm, DDPG, and Random algorithm increases slowly when the edge cloud's cache resource increases from 1 to 100 MHz. This is due to the fact that computing tasks are allocated to a collaborative edge computing offloading strategy with higher probability, and more computing tasks are offloaded to the edge cloud. Likewise, both MADDPG and DDPG-based algorithms manage the cache resource better in the edge cloud, compared to the random algorithm in which tasks are hybrid offloaded. Accordingly, the proposed JHORA-MADDPG and DDPG-based algorithms achieve a higher completion ratio than the random algorithm for different cache resources.

VIII. Conclusion

In this article, a joint hierarchical task offloading and resource allocation for the Cybertwin-based network architecture with edge computing and FL was proposed to achieve faster task processing and lower overhead while enhancing

the system's security and protecting data privacy. We first presented a hierarchical task offloading strategy for delay-tolerant and delay-sensitive missions, including collaborative edge computing offloading and hybrid alternating offloading patterns. Then, both hierarchical task offloading and resource allocation management were studied, and the MADDPG algorithm was presented to improve processing efficiency. Finally, we proposed the FL-based distributed model training approach for the MADDPG model to protect end devices' data privacy. Numerical results demonstrated that the proposed JHORA-MADDPG algorithm could effectively achieve faster task processing and lower overhead for better system processing efficiency and task completion ratio than other algorithms.

The MADDPG model and federation learning for Cybertwin-based network architectures are still an open issue. In the future, it is valuable to investigate the blockchain-enabled edge computing architecture. Our next step is to leverage FL and blockchain technologies to train the proposed model in a distributed way for data security and the training lightweight overhead.

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Infomat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[3] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart. 2018.

[4] N. Abbas,Y. Zhang, A.Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart. 2015.

[6] Q. Yu, J. Ren, H. Zhou, and W. Zhang, "A cybertwin based network architecture for 6G," in *Proc. 2nd 6G Wireless Summit (6G SUMMIT)*, Levi, Finland, 2020, pp. 1–5.

[7] S. Chen, Z. Pang, H. Wen, K. Yu, T. Zhang, and Y. Lu, "Automated labeling and learning for physical layer authentication against clone node and Sybil attacks in industrial wireless edge networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 2041–2051, Mar. 2021.

[8] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[9] Q. Yu, J. Ren, Y. Fu, Y. Li, and W. Zhang, "Cybertwin: An origin of next generation network architecture," *IEEE Wireless Commun.*, vol. 26, no. 6, pp. 111–117, Dec. 2019.

[10] M. Li, F. R. Yu, P. Si, and Y. Zhang, "Green machine-to-machine (M2M) communications with mobile edge computing (MEC) and wireless network virtualization," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 148–154, May 2018.

[11] S. Fu *et al.*, "Virtualization enabled multi-point cooperation with convergence of communication, caching, and computing," *IEEE Netw.*, vol. 34, no. 1, pp. 94–100, Jan./Feb. 2020.

[12] B. Ji, L. Sun, C. Li, C. Han, and H. Wen, "Throughput enhancement for wireless sensor network based on network allocation vector caching algorithms," *Ad Hoc Sens. Wireless Netw.*, vol. 40, pp. 49–72, Nov. 2018.

[13] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.

[14] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 146–152, Jun. 2017.

[15] Q.-V. Pham *et al.*, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020.

[16] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 4th Quart., 2019.

[17] W. Du *et al.*, "Approximate to be great: Communication efficient and privacy-preserving large-scale distributed deep learning in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11678–11692, Dec. 2020.

[18] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[19] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[20] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.

[21] S. Luo, X. Chen, Q. Wu, Z. Zhou, and S. Yu, "HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6535–6548, Oct. 2020.

[22] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey,"İ *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, Aug. 2020.

[23] T. Yang *et al.*, "Two-stage offloading optimization for energy–latency tradeoff with mobile edge computing in maritime Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5954–5963, Jul. 2020.

[24] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[25] M. Tang, L. Gao, and J. Huang, "Communication, computation, and caching resource sharing for the Internet of Things," *IEEE Commun. Mag.*, vol. 58, no. 4, pp. 75–80, Apr. 2020.

[26] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020.

[27] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, Jan. 2021.

[28] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.

[29] Y. He, C. Liang, F. R. Yu, and V. C. M. Leung, "Integrated computing, caching, and communication for trust-based social networks: A big data DRL approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, UAE, Feb. 2019, pp. 1–6.

[30] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12175–12186, Oct. 2020.

[31] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 46–51, Jun. 2020.

[32] D. Kwon, J. Jeon, S. Park, J. Kim, and S. Cho, "Multiagent DDPG-based deep learning for smart ocean federated learning IoT networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9895–9903, Oct. 2020.

[33] S. Yu, X. Chen, Z. Zhou, X. Gong and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultra-dense network," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2238–2251, Feb. 2021.

[34] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp 12313–12325, Dec. 2018.

[35] W. Hu and G. Cao, "Quality-aware traffic offloading in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3182–3195, Nov. 2017.

[36] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8577–8588, Oct. 2019.

[37] W. Zhan *et al.*, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.