Cooperative End-Edge-Cloud Computing and Resource Allocation for Digital Twin Enabled 6G Industrial IoT

Yuao Wang, Jingjing Fang, Yao Cheng, Hao She,
Yongan Guo, *Member, IEEE,* and Gan Zheng, *Fellow, IEEE*

***Abstract*—End-edge-cloud (EEC) collaborative computing is regarded as one of the most promising technologies for the Industrial Internet of Things (IIoT). It offers effective solutions for managing computationally intensive and delay-sensitive tasks efficiently. Indeed, achieving intelligent manufacturing in the context of 6G networks requires the development of efficient resource scheduling schemes. However, improving the quality of service and resource management in the face of challenges like time-varying physical operating environments of IIoT, task heterogeneity, and the coupling of different resource types is undoubtedly a complex task. In this work, we propose a digital twin (DT) assisted EEC collaborative computing scheme, where DT is utilized to monitor the physical operating environment in real-time and determine the optimal strategy, and the potential deviation between the real values and DT estimates is also considered. We aim to minimize the system cost by optimizing device association, offloading mode, bandwidth allocation, and task split ratio. Our optimization is constrained by the maximum tolerable latency of the task while considering both latency and energy consumption. To solve the collaborative computation and resource allocation (CCRA) problem in the EEC, we propose an algorithm with DT based on Multi-Agent Deep Deterministic Policy Gradient (MADDPG), where each user end (UE) in DT operates as an independent agent to determine the optimum offloading decision autonomously. Simulation results demonstrate the effectiveness of the proposed scheme, which can significantly improve the task success rate compared to benchmark schemes, while reducing the latency and energy consumption of task offloading with the assistance of DT.**

***Index Terms*—Collaborative computing, digital twin, industrial Internet of Things, resource allocation.**

I. Introduction

THE worldwide rollout of 5G technology has given rise to an unprecedented surge in intelligent devices and data traffic [1]. As a result, we are witnessing the emergence of groundbreaking applications and services like the Internet of Vehicles (IoV), Virtual Reality (VR), and Expanded Reality (ER) [2]. These advancements necessitate ultra-high data rates, ultra-low transmission latency, and seamless connectivity on a massive scale [3]. In response to the pressing demands stemming from these breakthroughs, both academia and industry have begun efforts to delve deeper into next-generation networks (6G) [4], [5]. Envisioned as a cornerstone of the future communications infrastructure, 6G is poised to play a crucial role in the forthcoming decade and beyond. In the impending era of 6G, a large number of Industrial Internet of Things (IIoT) end devices will be connected to the network, while the integration of artificial intelligence and 6G networks opens the door to intelligent manufacturing. Nevertheless, end devices in IIoT generate huge amounts of data during operation, but their computational and storage capacities are limited [6], and how to efficiently handle large-scale computationally intensive and delay-sensitive tasks has become an imperative problem.

Fortunately, mobile edge computing (MEC) offers the advantage of proximity, allowing intelligent computing and data processing capabilities to be deployed closer to the source of computational tasks generated by end devices [7], [8]. Meanwhile, task offloading plays a pivotal role in End-edge-cloud (EEC) systems. It facilitates the selective distribution of computing tasks across multiple tiers [9], which include user ends (UEs), edge servers (ESs), and cloud servers (CSs). By optimizing the allocation of computational resources across various tiers, latency is minimized and the overall performance of the system is enhanced [10]. More importantly, recent advancements in 6G have made it possible to orchestrate offloading strategies. Delay-sensitive tasks can be offloaded to ES to ensure real-time response, guaranteeing timely responsiveness. On the other hand, less delay-sensitive tasks can be processed in the CS, making full use of its abundant computing resources and maximizing efficiency. Despite the numerous advantages of introducing MEC into communication networks, designing efficient collaborative computing solutions remains a tricky challenge given the heterogeneity of end devices and ESs, the dynamic complexity of EEC networks, and the coupling between resources in IIoT.

An excellent digitization solution for intelligent manufacturing is provided by digital twin (DT), an emerging digital mapping technology [11]. DT empowers real-time monitoring of physical entities and processes, serving as a foundation for essential simulations, validations, predictions, and optimizations [12], [13]. All types of devices, networks, and computational processes in the IIoT can be digitally represented in the virtual world, thereby paving the way for intelligent task offloading

This work was supported in part by the Frontier Leading Technology Basic Research Project of Jiangsu Province BK20202001, in part by the Agricultural Science and Technology Independent Innovation Fund of Jiangsu Province CX (22) 3116, and in part by the Modern Agricultural Machinery Equipment and Technology Demonstration Project of Jiangsu Province NJ2022-13.

Yuao Wang, Jingjing Fang, Yao Cheng, Hao She, Yongan Guo, are with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 1220013721@njupt.edu.cn, blingblingfang@163.com, 2249661933@qq.com, hao.s@foxmail.com, guo@njupt.edu.cn).

Gan Zheng is with the School of Engineering, University of Warwick, CV47AL Coventry, U.K. (e-mail: gan.zheng@warwick.ac.uk).

and resource allocation [14], [15]. By combining MEC and DT, the overall network and resource information (such as bandwidth occupation, computing resource utilization, cache space, etc.) can be captured in a highly accurate and real-time manner [16], [17]. This information can then be analyzed to change the network configuration [18] and resource allocation to maximize the protection of service quality. However, there are still many open challenges in the integration of DT with EEC collaboration.

A. Related Works and Motivation

Existing research in MEC focuses on designing efficient offloading schemes [19] and resource management [20], [21], aiming to reduce task latency [22] and improve energy efficiency [23]. Dai et al. proposed a cooperative offloading solution for the IIoT, where computationally-intensive tasks can be offloaded either to ES or to IIoT devices via D2D links [19]. Li et al. researched the offloading optimization problem for multi-user delay-sensitive tasks and set different objectives for the cost and revenue of the tasks [22]. Malik et al. presented an integrated approach to minimize energy consumption by adjusting the transmitting power [23].

Additionally, resource allocation for EEC hierarchical [24] computing offloading [25], [26] has drawn a lot of interest. Chen et al. proposed a service placement and offloading strategy based on game theory [27]. Wu et al. proposed a blockchain-enabled edge cloud collaboration scheme [28], which ensures secure task offloading while reducing latency using Lyapunov optimization theory. However, multilayer task offloading usually jointly optimizes communication and computational resources and is quite difficult to solve for decisions. Using traditional optimization methods, only approximate optimal solutions can be obtained. Furthermore, motivated by the powerful modeling capabilities of Markov Decision Process (MDP) for dynamic EEC [29], [30], a number of research works have used Deep Reinforcement Learning (DRL) techniques for task offloading and resource cheduling [31]–[36]. Hazra et al. designed a computational offloading framework for industrial networks, which introduces a policy-based reinforcement learning (RL) technique to optimize the cost of devices [31]. For the issue of scarce edge computing resources, Sun et al. suggested a joint offloading technique based on resource occupancy prediction [32]. Li et al. proposed a task offloading strategy with the objective of maximizing long-term offloading revenue, which jointly optimizes the transmission power allocation and the computational resource allocation [34]. Suzuki et al. formulated a task offloading problem in multi-cloud and multi-edge networks. A method based on Multi-agent deep reinforcement learning (MADRL) is also proposed to address the performance of RL [35].

While these studies have delved into the realm of EEC collaboration to enhance the quality of service, it is worth noting that task offloading and communication scheduling are intricately intertwined with resource management [37]. The ever-changing nature of industrial environments, the heterogeneity of tasks, and the unpredictability of processing capability have presented unparalleled challenges to collaborative computing in the IIoT. Numerous studies have demonstrated the efficacy of DT in facilitating task offloading decisions. By creating digital representations of physical systems, DT enables optimized resource allocation, thereby bolstering network performance [38]–[40], [43]–[48]. Huynh et al. proposed a URLLC-based DT architecture which simultaneously considers communication, computation, and storage resources. The problem of optimizing latency/reliability in DT-enabled virtual worlds was addressed [38]. Guo et al. presented a D2D-assisted DT edge network architecture and employed federated RL to train resource allocation policies, thereby enhancing network performance further [40]. Lin et al. proposed an incentive-based congestion control scheme for addressing stochastic demand in a DT edge network. The scheme effectively avoids long-term service congestion and achieves near-optimal revenue by providing incentives to users [44]. A DT framework for IIoT networks is proposed where DT for UAVs is constructed to support real-time offloading of tasks [47]. The above contributions have been dedicated to studying DT for realizing joint computation and communication resources allocation. However, the benefits of DT-aided cloud computing have not been explored. Hou et al. presented a hierarchical task offloading approach empowered by DT to cater to both delay-tolerant and delay-sensitive tasks. Moreover, they leveraged federated learning techniques to optimize the decision model [41]. However, it is crucial to acknowledge that the disparity between the digital realm and the physical world has not been factored into their model. Consequently, this oversight can potentially lead to significant deviations between system predictions and actual values [42], hence adversely impacting overall performance.

The current utilization of DT is limited to the end devices and ES, failing to fully harness the capabilities offered by cloud computing. Therefore, further investigation and research are imperative to bridge this gap and maximize the potential of EEC collaboration. In the realm of IIoT, there arises a pressing demand for a comprehensive solution to integrate DT and EEC for efficient resource allocation. Indeed, an optimal solution necessitates a meticulous consideration of the distinctive attributes and deviations that define the digital world. Equally crucial is the comprehensive accommodation of the dynamic and unpredictable nature inherent to IIoT environments. This critical integration holds exceptional significance, particularly in the context of 6G IIoT.

B. Contributions and Structure

Driven by the above background, we propose a joint communication and computation offloading scheme in MC networks with DT that takes into account all the above issues. We aim to minimize the system cost of task offloading by optimizing device associations, offloading mode, bandwidth allocation, and task split while considering latency and energy consumption. Furthermore, we propose DT-empowered algorithms for collaborative computation and resource allocation (CCRA), which are based on MADRL. The main contributions of this paper are summarized as follows.

- We propose a DT-assisted cooperative computing framework for task hierarchical offloading in EEC. The frame-

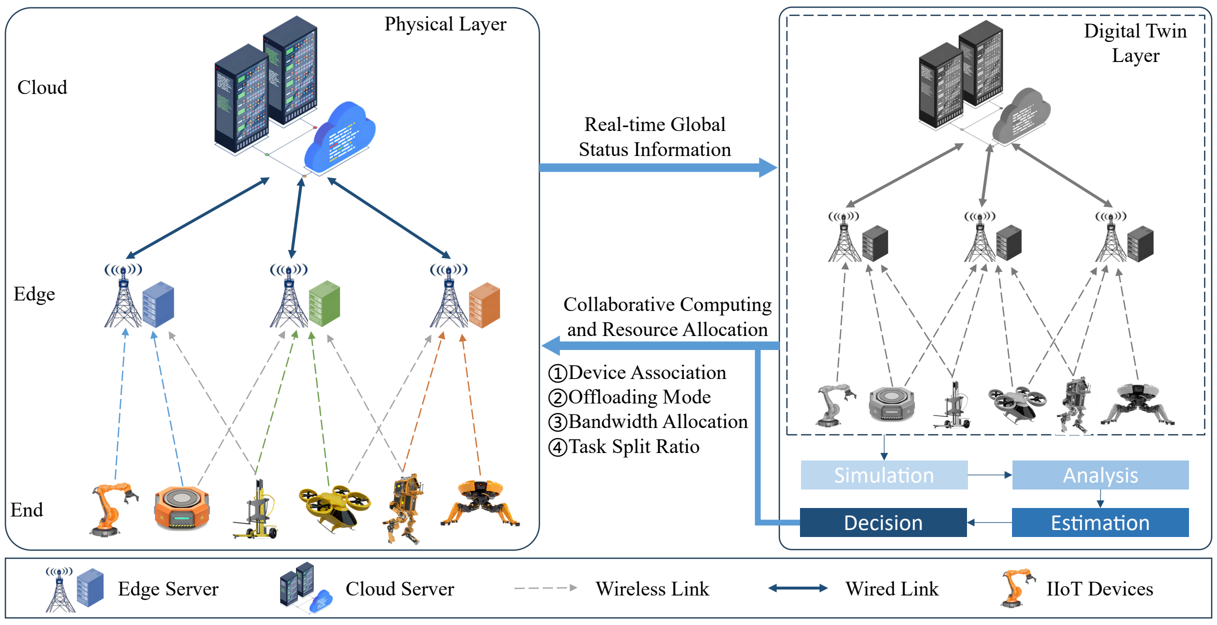

Fig. 1. DT enabled End-Edge-Cloud collaborative system model in IIoT.

work consists of two layers that provide flexible offloading modes and reduce the computational load on ESs. One layer is the UEs, ESs, and CS in IIoT, and the other layer is the digital representation of various entities. These DTs interact with their physical counterparts in real-time, allowing for seamless information exchange and decision-making. Furthermore, we consider the deviation between the physical system and DT estimation, and the regular pattern of change in the system due to the deviation is summarized.

- We formulate the cost minimization problem in a cooperative EEC system, which combines task latency and energy consumption, weighted according to their relative importance while addressing the allocation of computational and communication resources. Specifically, we achieve this by optimizing device association, offloading mode, bandwidth allocation, and task split, adhering to constraints imposed by task tolerance latency and the availability of resources.
- We utilize the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm to solve intricate optimization problems in CCRA. In particular, we deploy an agent within the DT of each UE, which obtains state information from the EEC for training. Notably, we address the challenge of hybrid action space by converting the discrete action of device association and offloading mode into a continuous probability.
- For comparative analysis, we conducted extensive simulations to explore the impact of different parameters on latency and energy consumption. The comprehensive simulation results demonstrate the effectiveness of our proposed scheme, as it significantly reduces system costs.

The remainder of this work is structured as follows. In Section II, we provide a description of the EEC collaboration model and present the problem formulation for DT-assisted IIoT. Section III details our proposed MADDPG-based algorithm, which addresses the CCRA problem while considering communication and computational resource limitations. In Section IV, we present numerical results and discussions of our findings. Lastly, Section VI concludes the study and outlines future plans.

II. SYSTEM MODEL AND PROBLEM FORMULATION

This paper builds an EEC collaborative computing model with the assistance of DT in the 6G-oriented IIoT. Based on this, in this section, we describe the system model, the communication model, and the computing model specifically. Finally, the objective problem is mathematically formulated in this section. The system parameters are illustrated in Table I.

A. System Model

As shown in Fig. 1, we propose an EEC collaborative computing system model containing DTs into two layers. The first layer is the physical layer, which represents the real physical entity devices, and the other layer is the DT layer, where the DT layer is a digital mapping of the physical devices.

The physical layer consists of M UEs, K ESs, and a CS c. The M UEs and K ESs are denoted by the sets $\mathcal{M} = \{1, 2, ..., M\}$ and $\mathcal{K} = \{1, 2, ..., K\}$, respectively.

The construction of DT layer necessitates the access to real-time global state information, including the operational status and requirements of the UEs, the available transmission and computation resources on the ESs and the CS, etc. DT of the global device is constructed on the edge side, denoted as [16]: DT $= \left\{\tilde{\mathcal{M}}, \tilde{\mathcal{K}}, \tilde{c}\right\}$

At the DT layer, task offloading options and offloading ratios are analyzed and estimated, and the DT serves as a decision maker to make offloading decisions by considering the system cost globally. Meanwhile, the DT also plays the role of a controller, delivering decisions to the physical layer and controlling the tasks to be carried out in the physical layer, so as to complete the computation and resource allocation in the physical layer. The DT layer continuously interacts with the physical layer to transfer information, including the global status of the physical layer and decisions by the DT layer. Through the construction of the DT, it realizes the collection of global information and the control of equipment operation states, and better utilization of global resources.

It is assumed that a computationally intensive task $S_m = \{D_m, C_m\}$ is generated at UE m, where D_m denotes the data volume of the task and C_m denotes the total number of CPU cycles required to execute this task independently. Due to the limited computing capabilities of the UEs to carry out the computation task, it is necessary to offload part of the task to an optional ES k or the CS c at a certain ratio, of which this paper will be divided into two parts: the task split and the offloading decisions.

In the task split section, we assume that the task is divided into two non-intersecting parts at a certain ratio, and the task processed locally on the UE m is defined as $S_{m,l} = \{D_{m,l}, C_{m,l}\}$, where $D_{m,l}$ denotes the amount of locally processed task data and $C_{m,l}$ denotes the total number of CPU cycles locally required to execute this task independently. The part of the task that needs to be offloaded to the ES k or the CS c is represented as $\gamma_m D_m$, by where γ_m is defined as the task splitting ratio, which satisfies $0 \leq \gamma_m \leq 1$ and $D_{m,l} = (1 - \gamma_m) D_m$.

In the offloading decisions section, we consider it in two steps:

1) Device association: The task is either handled by ES or CS. If the ES k has enough computing resources to handle the part of the task that needs to be offloaded by the UE m, the ES k is allowed to receive the computational task from the UE m, and define the binary variable for device association $\xi_{m,k} = 1$, otherwise $\xi_{m,k} = 0$. Each UE could only be associated with one ES at a time for task offloading,i.e.,

$$\sum_{k \in \mathcal{K}} \xi_{m,k} \leq 1, \forall m \in \mathcal{M}. \tag{1}$$

2) Offloading mode selection: If the computing resources of the current ES k are insufficient for processing, the ES k can be regarded as a relay to continue offloading the task to the CS c, and define the binary variable for offloading target selection $\xi_{m,c} = 1$ at this time, otherwise $\xi_{m,c} = 0$.

In summary, we denote the two offloading modes as:

UE to ES offloading: $\xi_{m,k} = 1, \xi_{m,c} = 0$. UE to CS offloading: $\xi_{m,k} = 1, \xi_{m,c} = 1$.

B. Communication Model

The communication process is usually divided into uplink communication and downlink communication. The uplink communication delay mainly includes the wireless communication delay from the UE to the ES and the wired communication delay from the ES to the CS. The downlink generally transmits small return data, so that the downlink communication delay is usually considered to be negligible.

TABLE I
THE ILLUSTRATION OF MAIN NOTATIONS.

Notations	Discriptions
$\mathcal{M}, \mathcal{K}$	Set of UEs and ESs
$\tilde{\mathcal{M}}, \tilde{\mathcal{K}}, \tilde{c}$	The DT of the UEs, the ESs and the CS
S_m	Task generated at UE m
D_m	Data size of the task S_m
$D_{m,l}, D_{m,k}, D_{m,c}$	Data size of the task processed locally, by the ES and by the CS
C_m	Total number of CPU cycles required to execute the task S_m
γ_m	Task offloading ratio factor
$\xi_{m,k}, \xi_{m,c}$	Binary variable for device association and offloading target selection
B	Total bandwidth of all the servers
$\lambda_{m,k}$	Bandwidth allocation ratio
R^{up}	Maximum communication rate of uplink wireless communication
P^t	Transmission power
$g_{m,k}$	Channel gain
${\sigma_k}^2$	The Gaussian white noise power
T^{com}	Communication delay
E^{com}	Communication energy consumption
$\tilde{f}_{m,l}, \tilde{f}_{m,k}, \tilde{f}_{m,c}$	Estimated CPU computing frequency available to the UE m by the UE m, the ES k and by the CS c in DT
$\Delta f_{m,l}, \Delta f_{m,k}, \Delta f_{m,c}$	CPU computing frequency deviation between the estimation and reality by the the UE m, the ES k and by the CS c
q_m	Effective switching capacitance of the local computing chip
e_k	Energy consumption of ES k processing one CPU cycle
T^{cmp}	Computing delay
E^{cmp}	Computingn energy consumption
T_m^{total}	Total latency of UE m
E_m^{total}	Total energy consumption of UE m
ω_1, ω_2	Latency weight factor and energy consumption weight factor
Q^{system}	System cost
T_m^{max}	Maximum tolerable time for the UE m

1) UE to ES offloading: Multiple UEs share the same channel resource and offload to the ES k at the same time. The set of UEs associated with the ES k to be $\mathcal{M}_k = \{1, 2, ..., M_k\}, M_k \in \mathcal{M}$. When the UE m offloads a task $D_{m,k} = \gamma_m D_m$ to the ES k, the bandwidth allocated to the UE m by the ES k is B_m, i.e.,

$$B_m = \lambda_{m,k} B \tag{2}$$

where B is the total channel bandwidth of the ES k and $\lambda_{m,k}$ is the bandwidth allocation ratio. ESs operate in different frequency bands and there is no interference between ESs as well as between UEs [54]. The maximum communication rate of uplink wireless communication from the UE m to the ES

k can be expressed as

$$R_{m,k}^{up} = B_m \log_2 \left(1 + \frac{P_m^t g_{m,k}}{{\sigma_k}^2}\right) \tag{3}$$

where P_m^t is the transmission power of the UE, $g_{m,k}$ is the channel gain, and ${\sigma_k}^2$ is the Gaussian white noise power. The task uplink communication delay of the UE m is expressed as:

$$T_{m,k}^{com} = \frac{D_{m,k}}{R_{m,k}^{up}}. \tag{4}$$

Hence, the communication energy consumption of the UE m is computed as

$$E_{m,k}^{com} = P_m^t T_{m,k}^{com}. \tag{5}$$

2) UE to CS offloading: In this case, the uplink communication delay contains the wireless communication delay and wired communication delay. The wired communication delay can be approximated simply as:

$$T_{k,c}^{com} = \frac{D_{m,c}}{\nu} \tag{6}$$

where $D_{m,c}$ is the task which is offloaded from the UE m to the CS c via the ES k and ν is the wired transmission rate between the ES k and the CS c [42]. It is worth noting that ν is set as a constant, and the links between each ES and CS are independent of each other, without mutual interference. The delay $T_{k,c}^{com}$ only depends on the size of the data.

As a result, the communication delay required for UE m to offload tasks to the CS c is given by

$$T_{m,c}^{com} = T_{m,k}^{com} + T_{k,c}^{com}. \tag{7}$$

Hence, the transmission energy consumption of the ES k can be expressed as

$$E_{k,c}^{com} = P_k^t T_{k,c}^{com}. \tag{8}$$

The total communication energy consumption is given by

$$E_m^{com} = E_{m,k}^{up} + \xi_{m,c} E_{k,c}^{up}. \tag{9}$$

C. Computing Model

1) Local computation: The estimated computing latency for local processing of the UE m can be written as

$$\tilde{T}_{m,l}^{cmp} = \frac{(1-\gamma_m) C_m}{\tilde{f}_{m,l}} \tag{10}$$

where $\tilde{f}_{m,l}$ is the estimated CPU computing frequency available to the UE m in the DT. Assuming that the difference between the real computing resources of the UE m in the physical entity and the estimated computing resources of the UE m in the DT has been obtained in advance, and defining the difference as $\Delta f_{m,l}$, the computing delay deviation is given as follows [49]

$$\Delta T_{m,l}^{cmp} = \frac{\Delta f_{m,l} (1-\gamma_m) C_m}{\tilde{f}_{m,l} \left(\tilde{f}_{m,l} - \Delta f_{m,l}\right)}. \tag{11}$$

Then the real local computing delay [35] can be expressed as

$$T_{m,l}^{cmp} = \tilde{T}_{m,l}^{cmp} + \Delta T_{m,l}^{cmp}. \tag{12}$$

The energy consumption for locally processing the computing task is given by:

$$E_{m,l}^{cmp} = (1-\gamma_m) q_m C_m \left(\tilde{f}_{m,l} - \Delta f_{m,l}\right)^2 \tag{13}$$

where q_m is the effective switching capacitance of the local computing chip [51].

2) ES computation: The estimated computing delay at the ES k is given as

$$\tilde{T}_{m,k}^{cmp} = \frac{\gamma_m C_m}{\tilde{f}_{m,k}} \tag{14}$$

where $\tilde{f}_{m,k}$ is the estimated CPU computing frequency assigned to the UE $\tilde{m}$ by the ES $\tilde{k}$ in the DT. Similarly, the difference is $\Delta f_{m,k}$. Then the computational delay deviation can be expressed as

$$\Delta T_{m,k}^{cmp} = \frac{\Delta f_{m,k} \gamma_m C_m}{\tilde{f}_{m,k} \left(\tilde{f}_{m,k} - \Delta f_{m,k}\right)}. \tag{15}$$

The real computing delay of the ES k is written as

$$T_{m,k}^{cmp} = \tilde{T}_{m,k}^{cmp} + \Delta T_{m,k}^{cmp}. \tag{16}$$

The energy consumption of the ES k for processing the computing tasks can be expressed as

$$E_{m,k}^{cmp} = \gamma_m e_k C_m \left(\tilde{f}_{m,k} - \Delta f_{m,k}\right)^2 \tag{17}$$

where e_k represents the energy consumption of ES k processing one CPU cycle [50].

3) CS computation: The estimated computing delay at the CS c is expressed as:

$$\tilde{T}_{m,c}^{cmp} = \frac{\gamma_m C_m}{\tilde{f}_{m,c}} \tag{18}$$

where $\tilde{f}_{m,c}$ is the estimated CPU computing frequency assigned to the UE $\tilde{m}$ by the CS $\tilde{c}$ in the DT. Similarly, the difference is $\Delta f_{m,c}$. Then the computational delay deviation is given by

$$\Delta T_{m,c}^{cmp} = \frac{\Delta f_{m,c} \gamma_m C_m}{\tilde{f}_{m,c} \left(\tilde{f}_{m,c} - \Delta f_{m,c}\right)}. \tag{19}$$

The real computing delay of the CS c can be expressed as

$$T_{m,c}^{cmp} = \tilde{T}_{m,c}^{cmp} + \Delta T_{m,c}^{cmp}. \tag{20}$$

CS c doesn't require any consideration of computational energy consumption [52] due to the assumption of unlimited storage resources.

Therefore, the total computing energy consumption of the UE m is given by.

$$E_m^{cmp} = E_{m,l}^{cmp} + (1-\xi_{m,c}) E_{m,k}^{cmp}. \tag{21}$$

In this paper, we define the offloading latency as the maximum value that depends on the sum of the computation and communication delays of each part of the task due to the parallelism of computation and communication, which can be denoted as

$$\begin{aligned} T_{m,l}^{total} &= T_{m,l}^{cmp} \\ T_{m,k}^{total} &= T_{m,k}^{com} + (1-\xi_{m,c}) T_{m,k}^{cmp} \\ T_{m,c}^{total} &= \xi_{m,c} \left(T_{m,c}^{com} + T_{m,c}^{cmp}\right) \end{aligned} \tag{22}$$

$$T_m^{total} = \max\left\{T_{m,l}^{total}, T_{m,k}^{total}, T_{m,c}^{total}\right\}. \tag{23}$$

The offloading energy consumption is expressed as

$$E_m^{total} = E_m^{com} + E_m^{cmp}. \tag{24}$$

D. Problem Formulation

In this paper, we define the system cost as:

$$Q^{system}(T_m^{total}, E_m^{total}) = \sum_{m=1}^{M}\left(\omega_1 T_m^{total} + \omega_2 E_m^{total}\right) \tag{25}$$

where ω_1, ω_2 are the latency weight factor and the energy consumption weight factor, respectively, which satisfy $0 \leq \omega_1 \leq 1$,$0 \leq \omega_2 \leq 1$ and $\omega_1 + \omega_2 = 1$.

An optimization problem in this paper is formulated to complete the problem optimization while satisfying the minimum cost of the task system. The problem is formulated as follows:

$$\begin{aligned} P1: \min_{\substack{\gamma_m,\lambda_m\\ \xi_{m,k},\xi_{m,c}}} \quad & Q^{system}\left(T_m^{total}, E_m^{total}\right) && (26)\\ s.t. \quad & T_m^{total} \leq T_m^{\max}, \forall m \in \mathcal{M} && (26a)\\ & 0 \leq \gamma_m \leq 1, \forall m \in \mathcal{M} && (26b)\\ & \sum_{k\in\mathcal{K}} \xi_{m,k} \leq 1, \forall m \in \mathcal{M} && (26c)\\ & 0 \leq \lambda_m \leq 1, \forall m \in \mathcal{M}_k && (26d)\\ & \sum_{m\in\mathcal{M}_k} \lambda_m = 1 && (26e) \end{aligned}$$

where $T_m^{\max}$ is the maximum tolerable time for the UE m to yield a task, and if this time value is exceeded, the task is labeled as a failure. Constraint (26a) indicates that the total latency for completing the task must be within the tolerable range. Constraint (26b) describes the task splitting ratio requirement. Constraint (26c) denotes that the UE can only be associated with one ES for task offloading. Constraint (26d) depicts the bandwidth split ratio requirement.

The problem $P1$ consists of continuous decisions about the task split ratio γ_m, bandwidth allocation λ_m and discrete decisions about device association $\xi_{m,k}$ and offloading mode selection $\xi_{m,c}$. Meanwhile, the states of UE, ES, and CS are highly dynamic and dependent so some uncertain variables (i.e., task size and channel conditions) can have an impact on energy consumption and latency. Additionally, given the large solution space for decision making, it is a great challenge for traditional optimization methods to obtain the optimal policy. In the next section, we will propose a MADRL-based approach to deal with hybrid actions in collaborative computing to output CCRA strategies in an online learning manner.

III. MADDPG-Based Algorithm for CCRA with DT

In this section, we transform the optimization problem, P1, into a multi-agent MDP. Subsequently, we put forward a MADDPG-based CCRA algorithm to tackle the problem. We improve upon the MADDPG algorithm to make it more suitable for solving DT-assisted CCRA problems. This is because the MADDPG algorithm is applicable for solving problems with multiple interacting agents, where each agent can share experience information while learning, which helps to learn the globally optimal policy.

A. MDP Model of CCRA

In RL, agents optimize their strategies automatically through interaction with the environment. By employing deep neural networks (DNN) to train learning models, faster learning speeds and improved performance can be achieved. DRL has found widespread applications in solving various complex sequential decision-making problems in task offloading. However, traditional single-agent approaches suffer from scalability issues, given the heterogeneous multidimensional resources and dynamic environments. Moreover, there is a risk of a system-wide breakdown in the event of a central failure.

To address the aforementioned challenges, we propose a solution that leverages DT for MADRL. Within the DT layer, different types of DTs can interact with one another, exchanging valuable information. Each DT of UE (DT-UE) serves as an independent agent, utilizing global information to decide strategies for collaboration and resource allocation. By leveraging this approach, UEs are able to transcend the limitations imposed by restricted computing resources in the physical layer, thus enabling to achieve of autonomous decision-making.

In DT-assisted EEC system, the DT-UE takes the responsibility of making a series of decisions. These decisions encompass critical factors, such as determining the optimal device association, offloading mode, bandwidth allocations, and task split ratios, with the primary objective of minimizing the total system cost. It is important to acknowledge that the decisions made by the DT-UE hold a profound influence over the state of the environment within the system. As a result, the overall system cost not only depends on the current state of the system environment but also on the collaborative actions taken by all DT-UEs. Moreover, it should be noted that the state of the system environment and the actions taken by the DT-UEs in the previous time slot can trigger a transition to a new state. Indeed, in order to effectively represent the decision-making process of CCRA, we transform the optimization problem into a multi-agent MDP.

Each DT-UE acts as an agent to learn its strategy and collectively they obtain the minimum total system cost. The MDP of M agents can be represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, where $\mathcal{S}$ describes the state of the environment. Taking advantage of the deployment of agents in the DT layer, information exchange can be facilitated among UEs, ESs, and CS. Each agent's observation contributes to the composition of the state space. $\mathcal{A}$ is the set of all agent execution actions. $\mathcal{R}$ is the set of individual agent reward functions, defined as the revenue from executing the action in the current state. $\mathcal{P}$ is denoted as the probability of transferring from the current state to the next state. In the following, we describe the key elements in MDP, including the state space, action space, and reward function.

1) State Space $\mathcal{S}$: In DTs, the environment is fully observed and hence the state of the environment consists of observations of individual agents. The set of states in time slot t is described as

$$\boldsymbol{s}(t) = \{o_1(t), ..., o_m(t), ..., o_M(t)\} \tag{27}$$

where $o_m(t)$ denotes the observation of agent m, which includes

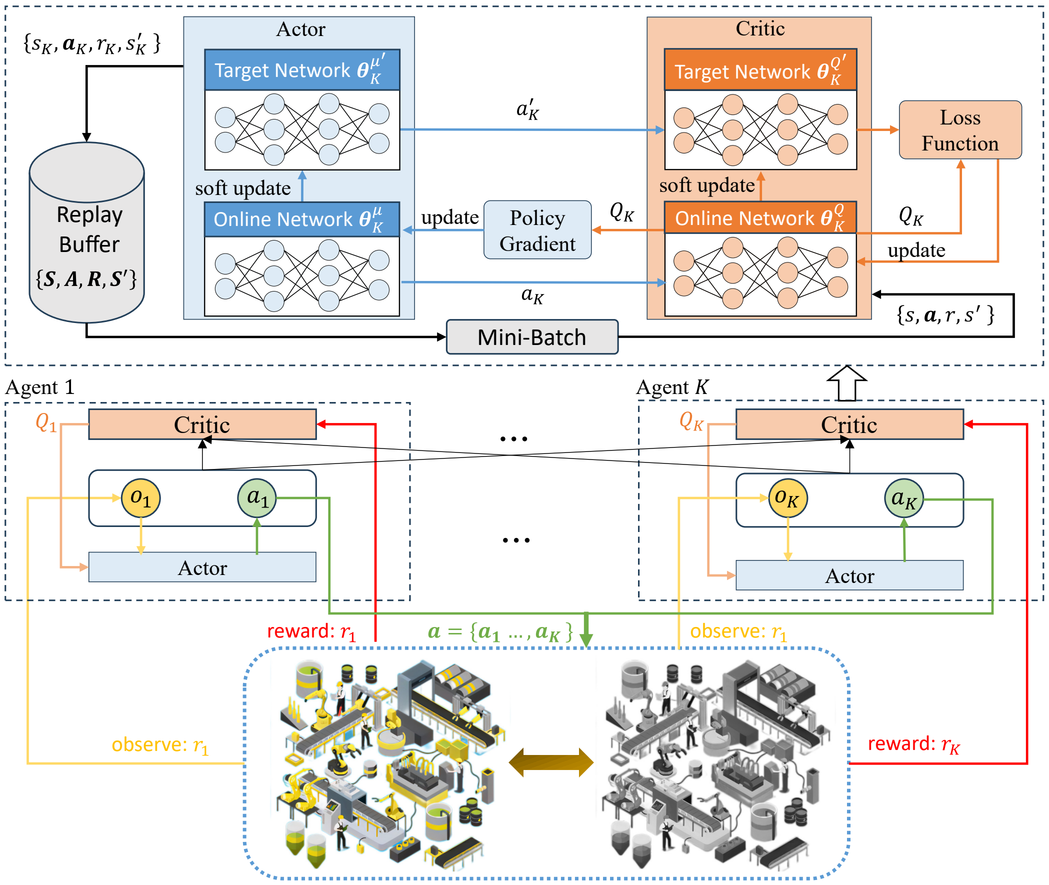
$\{s_K, \boldsymbol{a}_K, r_K, s_K'\}$
Actor
Target Network $\theta_K^{\mu'}$
soft update
Online Network θ_K^{μ}
Replay Buffer
$\{\boldsymbol{S}, \boldsymbol{A}, \boldsymbol{R}, \boldsymbol{S}'\}$
a_K'
update
Policy Gradient
Q_K
a_K
Critic
Target Network $\theta_K^{Q'}$
soft update
Online Network θ_K^{Q}
Loss Function
Q_K
update
$\{s, \boldsymbol{a}, r, s'\}$
Mini-Batch
Agent 1
Q_1
Critic
o_1
a_1
Actor
Agent K
Q_K
Critic
o_K
a_K
Actor
reward: r_1
$\boldsymbol{a} = \{\boldsymbol{a}_1 \dots, \boldsymbol{a}_K\}$
observe: r_1
observe: r_1
reward: r_K

Fig. 2. MADDPG-Based algorithm for CCRA with DT.

- The current UE m generates tasks $S_m = \{D_m, C_m\}$, from real-time interactions with the physical layer.
- The computational power of ES and CS, and the deviation in DT, respectively, $\tilde{f}_m, \Delta f_m, \tilde{f}_{m,k}, \Delta f_{m,k}$.
- The previous bandwidth allocation $\lambda_m(t-1)$ is available.
- The number of UEs connected to each ES $\mathcal{M}_k(t-1)$, this is fed back by the DT of ES.

2) Action Space $\mathcal{A}$: As in Eq. (26), the decisions made by the agent include the device association, offloading mode, bandwidth allocations, and task split ratios, then the action of the DT-UE m is described as

$$a_m(t) = \{\gamma_m(t), \lambda_m(t), \xi_{m,k}(t), \xi_{m,c}(t)\} \tag{28}$$

According to the constraints, the values of the four variables range from $\gamma_m(t) \in [0,1]$, $\lambda_m(t) \in [0,1]$, $\xi_{m,k}(t) \in \{0,1\}$, $\xi_{m,k}(t) \in \{0,1\}$. In addition, the size of the action increases with the growing number of UEs and ESs.

3) Reward Function $\mathcal{R}$: The agent takes actions based on the current system state, and after performing an action, it receives a reward. The reward reflects the desirability or quality of the action taken in that particular state. Agents should collaboratively minimize the total system cost while accomplishing the computational task. Hence, the reward function includes the negative value of cost $\left(\omega_1 T_m^{total} + \omega_2 E_m^{total}\right)$ for each DT-UE as a component. Additionally, if a task is not completed within the designated time, a penalty η is incorporated into the reward function. The reward for the agent m can be expressed as

$$\mathcal{R}_{\mathrm{m}}(t) = \begin{cases} -\left(\omega_1 T_m^{total} + \omega_2 E_m^{total}\right), & \text{tasks completed;} \\ -\left(\omega_1 T_m^{total} + \omega_2 E_m^{total}\right) + \eta, & \text{otherwise.} \end{cases} \tag{29}$$

B. MADDPG-Based Algorithm

To solve the above multi-agent MDP, we propose a MADDPG-based approach considering the high-dimensional hybrid action space of the task offloading optimization problem, shown in Fig. 2. The DDPG algorithm [52] is deployed on each DT-UE, which consists of an actor network $\mu_m(o_m; \theta_m^{\mu})$ with weight θ_m^{μ} and a critic network $Q_m(\boldsymbol{o}, \boldsymbol{a}; \theta_m^{Q})$ with weight θ_m^{Q}. The actor network receives the local observation o_m as input and generates the corresponding action $a_m(t) = \mu_m(o_m(t))$. The critic network is able to access the observations and actions of all the agents during the training phase in order to calculate the Q-value $Q_m(\boldsymbol{s}(t), \boldsymbol{a}(t))$, which is used to evaluate the performance of the actor network's output. Both the actor and critic networks in the DT-UE utilize target networks, denoted as $\mu_m^{'}(o_m; \theta_m^{\mu^{'}})$ and $Q_m^{'}(o, a; \theta_m^{Q^{'}})$, to update parameters θ_m^{μ} and θ_m^{Q}, respectively. This technique enhances training stability and convergence by periodically updating the target networks with the parameters from the actor and critic networks.

It is worth noting that actions $\gamma_m(t)$ and $\lambda_m(t)$ are continuous, while actions $\xi_{m,k}(t)$ and $\xi_{m,c}(t)$ are discrete values. Traditional DRL algorithms can only handle problems where actions are either continuous or discrete, but they cannot handle the hybrid action space. Ding et al. have demonstrated the feasibility of converting the discrete action space into a probability distribution [53]. We define the actor network as $\widehat{\mu}_m(o_m;\widehat{\theta}_m^{\mu})$, the output action $\widehat{a}_m(t) = \left\{\gamma_m(t), \lambda_m(t), \widehat{\xi}_{m,k}(t), \widehat{\xi}_{m,c}(t)\right\}$, where $\widehat{\xi}_{m,k}(t), \widehat{\xi}_{m,c}(t)$ are the probability distributions of the two discrete actions respectively. This means that we can use DDPG to solve single-agent MDP with discrete action spaces. Since MADDPG is a multi-agent extension of DDPG, it is effective for solving hybrid action space problems.

In the MADDPG algorithm, an experience replay mechanism is employed, allowing each agent to store the current experience tuple $(\widehat{\boldsymbol{a}}(t), \mathcal{R}(t), \boldsymbol{s}(t), \boldsymbol{s}(t+1))$ in the replay buffer $\mathcal{B}$ at each time step. For each DT-UE, we can sample a random mini-batch of $(\widehat{\boldsymbol{a}}_j, \boldsymbol{r}_j, \boldsymbol{s}_j, \boldsymbol{s}_j^{'})$ from $\mathcal{B}$. Random sampling in the training process breaks the correlation between sample data and reduces training oscillations. Afterward, the sampled experiences s_j is fed into the actor network, and the weights of the actor network are updated using the policy gradient strategy, i.e.,

$$\begin{aligned}&\nabla_{\theta_m^{\mu}} J(\mu_m)\\&= \frac{1}{N_b}\sum_{j=1}^{N_b}\left[\nabla_{\theta_m^{\mu}}\mu_m(s_j)\nabla_{a_m}Q_m(s_j,\widehat{a}_1,...,\widehat{a}_M)|_{\widehat{a}_m=\mu_m(s_j)}\right]\end{aligned} \tag{30}$$

where N_b is the size of the mini-batch, the actor network can be updated by minimizing the loss as below:

$$L\left(\theta_m^{\mu}\right) = -\frac{1}{N_b}\sum_{i=1}^{N_b} Q_m(s_j, a_1, ..., a_m, ..., a_M; \theta_m^Q). \tag{31}$$

Moreover, the critical network updates the parameters by minimizing the loss of the target Q-values,

$$L\left(\theta_m^{Q}\right) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left[y_m(j) - Q_m(s_j, \widehat{a}_1, ..., \widehat{a}_M; \theta_m^Q\right]^2 \tag{32}$$

where the update target $y_m(j)$ is calculated as,

$$y_m(j) = r_j + \varepsilon Q_m^{'}\left[s_j^{'}, \mu_1^{'}\left(s_j^{'};\theta_1^{Q^{'}}\right), ..., \mu_M^{'}\left(s_j^{'};\theta_M^{Q^{'}}\right); \theta_m^{Q^{'}}\right] \tag{33}$$

where ε is the discount factor.

The DT-assisted MADDPG method for CCRA is summarized in Algorithm 1. First, we initialize the parameters of the four neural networks and replay buffer. In each episode, DT-UE obtains the observation $o_m(t)$, which is then input to the actor network, and the output consists of the bandwidth allocation ratio $\gamma_m(t)$, the task split ratio $\lambda_m(t)$, and the probability distributions of the device association $\widehat{\xi}_{m,k}(t)$ and the offloading mode $\widehat{\xi}_{m,c}(t)$. Sampling $\widehat{\xi}_{m,k}(t)$ and $\widehat{\xi}_{m,c}(t)$, the current probabilities and decisions are mapped as,

$$\xi_{m,k}(t) = \begin{cases}1, & \widehat{\xi}_{m,k}(t) \in (0.5, 1]\\ 0, & \widehat{\xi}_{m,k}(t) \in [0, 0.5]\end{cases} \tag{34}$$

Algorithm 1 MADDPG-Based Algorithm for CCRA with DT.

```
1:  Initialize replay buffer $\mathcal{B}$ for each DT-UE
2:  Initialize actor network $\{\theta_m^{\mu'}\}_{m=\{1,...,M\}}$ and target actor
    network $\{\theta_m^{\mu}\}_{m=\{1,...,M\}}$ for each DT-UE, respectively
3:  Initialize critic network $\{\theta_m^{Q}\}_{m=\{1,...,M\}}$ and target critic
    network $\{\theta_m^{Q'}\}_{m=\{1,...M\}}$ for each DT-UE, respectively
4:  for each episode do
5:      Observe initial state $s_0$
6:      for each time $t$ do
7:          for each DT-UE $m$ do
8:              DT-UE $m$ observing environment and get $o_m(t)$
9:              DT-UE $m$ select action $\widehat{a}_m(t) = \widehat{\mu}_m(o_m;\widehat{\theta}_m^{\mu})$
10:             Sampling based on probability distribution
11:             Obtain reward $\mathcal{R}_t$ and next environment state $\boldsymbol{s}_{t+1}$
                by $\boldsymbol{s}_t$ and $\boldsymbol{a}_t$
12:         end for
13:         if Replay buffer has space then
14:             Store experience $(\boldsymbol{a}_t, \mathcal{R}_t, \boldsymbol{s}_t, \boldsymbol{s}_{t+1})$ in replay buffer
15:         else if Replay buffer is full then
16:             Replacing the earliest set of experiences in the re-
                play buffer with $(\boldsymbol{a}_t, \mathcal{R}_t, \boldsymbol{s}_t, \boldsymbol{s}_{t+1})$ in replay buffer
17:         end if
18:     end for
19:     for each DT-UE $m$ do
20:         Sample several random mini-batches of $N_b$ experi-
            ence tuples from replay buffer $\mathcal{B}$.
21:         Update the critic network by minimizing critic loss
            $L\left(\theta_m^{Q}\right) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left[y_m(j) - Q_m(s_j, \widehat{a}_1, ..., \widehat{a}_M; \theta_m^Q\right]^2$
22:         Update the actor network by minimizing critic loss
            $L\left(\theta_m^{\mu}\right) = -\frac{1}{N_b}\sum_{i=1}^{N_b} Q_m(s_j, a_1, ..., a_M; \theta_m^Q)$
23:         Soft updates for actor target network and critic target
            network respectively:
24:         $\theta_m^{\mu'} = \tau\theta_m^{\mu} + (1-\tau)\theta_m^{\mu'}$
25:         $\theta_m^{Q'} = \tau\theta_m^{Q} + (1-\tau)\theta_m^{Q'}$
26:     end for
27: end for
```

$$\xi_{m,c}(t) = \begin{cases}1, & \widehat{\xi}_{m,c}(t) \in (0.5, 1]\\ 0, & \widehat{\xi}_{m,c}(t) \in [0, 0.5]\end{cases} \tag{35}$$

After the execution of the action, the environment will transition to the next state $s(t+1)$, and all DT-UEs will get a reward $\mathcal{R}(t)$ accordingly and new observation $o(t)$. For those agents who do not complete the task, they will receive a penalty. If the replay buffer is not full, the experience $(a(t), \mathcal{R}(t), s(t), s(t+1))$ of that phase will be stored directly, otherwise the earliest batch of experience will be replaced. Subsequently, DT-UE will use mini-batch sampling in the replay buffer to update the parameters of the neural network. The parameters of the actor network and the critical network are updated by minimizing the loss function. In addition, the parameters of the two target networks are updated using soft updating, which are defined as,

$$\begin{aligned} \theta_m^{\mu'} &= \tau\theta_m^{\mu} + (1-\tau)\theta_m^{\mu'} \\ \theta_m^{Q'} &= \tau\theta_m^{Q} + (1-\tau)\theta_m^{Q'} \end{aligned} \tag{36}$$

where τ is the update rate.

After the training is completed, the critic network is not needed and the decisions of the DT-UE will act on the real environment. The agent m makes decisions based on observation a_m and real-time transmits the three actions a_m of device association, offloading mode, and task split to IID and other DTs in the DT layer. The DT of ES transmits the action instruction of bandwidth allocation to the corresponding ES in the physical layer. Then the changes in the physical layer will be fed back to the DT-UE in real time to enter the next state.

Algorithm 2 Execution Phase Algorithm at Each DT-UE.

```
1: for each episode do
2:     for each time $t$ do
3:         Observe initial state $o_m$
4:         DT-UE $m$ select action $\widehat{a}_m(t) = \widehat{\mu}_m(o_m; \widehat{\theta}_m^{\mu})$
5:         Sampling based on probability distribution
6:         Sending commands to the physical layer
7:         Real-time feedback of environmental changes to the
           DT layer and the rest of the DTs
8:     end for
9: end for
```

IV. Simulation and numerical results

In this section, we evaluate the performance of the proposed scheme by extensive simulations. Firstly, we present the details of the simulation environment and algorithm parameters. Secondly, to demonstrate the superiority of the proposed scheme, we carry out a comparative analysis with several benchmark schemes. Finally, we analyze the influence of the deviation between the real values and estimations on the system performance.

A. Simulation Settings

We consider an IIoT scenario based on EEC. The parameterization of the system is based on the work [42], [50].

We distribute all the UEs and ESs uniformly across the region, and we assume that one CS and multiple ESs provide computation services to several UEs. In the system, the bandwidth of each ES is set as $B = 20$MHz, and the transmission rate between ES and CS is set to $v = 100$Mbps. For UE m, the transmission power $P_m^t = 0.5$W, the channel gain $g_{m,k} = -20$dB, and the Gaussian white noise power $\sigma_k{}^2 = 2 \times 10^{-12}$W. The effective switching capacitance of the local computing is set as $q_m = 10^{-27}$, and the estimated CPU computing frequency of UE m is $\tilde{f}_m = 0.1$GHz. The parameters for ES k, the estimated CPU computing frequency $\tilde{f}_{m,k} = 5$GHz, and the energy consumption for processing one CPU cycle is set to $e_k = 10^{-9}$. For CS, the estimated CPU computing frequency is set as $\tilde{f}_{m,c} = 100$GHz. We consider that both latency and energy consumption are significant factors in IIoT, so the weights $\omega_1 = \omega_2 = 0.5$. Both the simulation environment and the algorithm are executed on Ubuntu 20.04.5, using Python 3.8.13 and PyTorch 1.12.1. In our proposed algorithm framework, the Actor and Critic networks include two hidden layers, both of size 400, and the parameter settings of the algorithm are summarized in Table II.

TABLE II
Algorithm Parameters

Parameters	Value
Soft update rate τ	0.0003
Mini-batch N_b	64
Discount factor ε	0.99
Replay buffer size $\mathcal{B}$	10000
Penalty η	10

B. Performance Evaluation of Proposed Scheme

In this subsection, we aim to substantiate the advantages of the cooperative EEC framework and the MADDPG-Based Algorithm with DT, we compare our proposed solution with the following five benchmark solutions as follows:

- Double deep Q-Learning (DDQN): We implemented a DDQN-based offloading method for collaborative EEC systems. The bandwidth allocation and task splitting ratios were discretized into 32 levels. The settings of the algorithm parameters, such as the discount factor and the size of the replay buffer, are the same as in this paper.
- No DT: In this scheme, DT is not utilized. As a result, without considering the system state, UEs randomly assign a portion of their tasks to ESs or CS that they can establish a connection. Moreover, due to the absence of essential information exchanges, UEs use greedy strategies to compete with one another for the limited system bandwidth.
- Local Computing: All tasks are executed locally at the UE, and algorithms are no longer needed to output decisions on task split and resource allocation.
- Edge-End (EE) only: In this approach, we exclude the involvement of CSs. Instead, all tasks within the system are collaboratively computed by the UEs and ESs using the MADDPG-based algorithm. Notably, the discrete action space primarily focuses on device associations, while the continuous actions encompass bandwidth allocation and task segmentation.
- Cloud Computing: ESs in this scheme solely act as relay nodes and lack computational capability. All tasks are executed by UEs and one CS.

Fig. 3 shows the convergence performance of the proposed DT-assisted scheme, DDQN algorithm, and benchmark schemes in terms of average system cost. The MADDPG-based algorithm for CCRA with DT proposed by us requires the agents to continuously make decisions and interact with the environment in each epoch, and update the network model

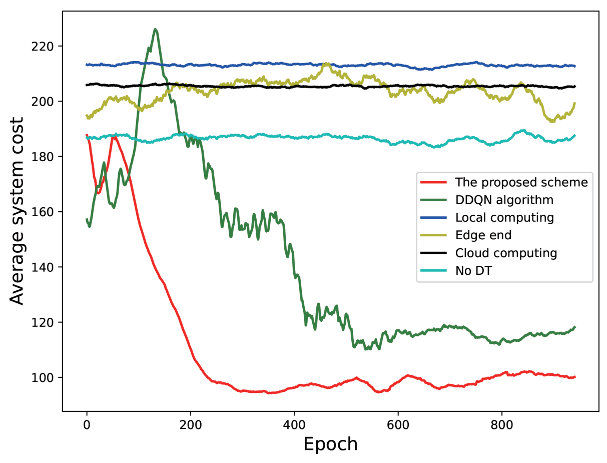

Fig. 3. The average system cost decreases with the increase of training epoch.

based on environmental feedback. It can be seen that the average system cost of our proposed algorithm and the DDQN algorithm continuously decreases with the increase in training epochs. The proposed algorithm starts to converge after 230 episodes, while the DDQN algorithm starts to converge after 600 epochs and eventually becomes stable. Therefore, our proposed algorithm converges faster than the DDQN method. In addition, the final average cost of our proposed solution is lower than the DDQN algorithm and other solutions, which validates the effectiveness of our approach.

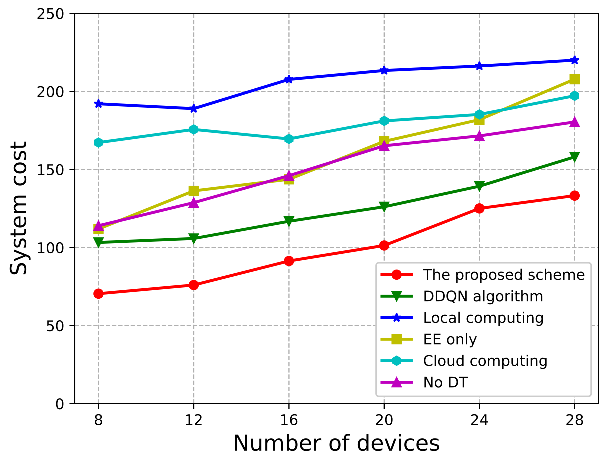

Fig. 4. Impact of number of UEs on system cost. There are 3 ESs.

Fig. 4 depicts the correlation between the system cost and the number of UEs, while keeping the number of ESs fixed at 4. It is evident that as the number of UEs rises, the system cost escalates across the six schemes to varying extents. This can be attributed to the increased computational tasks, consequently necessitating higher consumption of computational and communication resources. Notably, our proposed scheme exhibits the most cost-effective solution, being 33.7% lower than the average cost of the other benchmark schemes at a device count of 28. Conversely, Local Computing showcases the highest system cost due to the inherent limitations of its computational resources. Local Computing incurs the highest system cost due to its limited computing resources. In contrast, 'EE only' offers reduced communication and computation delay compared to both Local Computing and Cloud Computing. However, beyond a threshold of 20 UEs, the system cost of 'EE only' surpasses that of Cloud Computing. This can be attributed to certain ESs being overloaded with excessive computational tasks, thereby resulting in prolonged processing delays, consequently driving up the overall system cost.

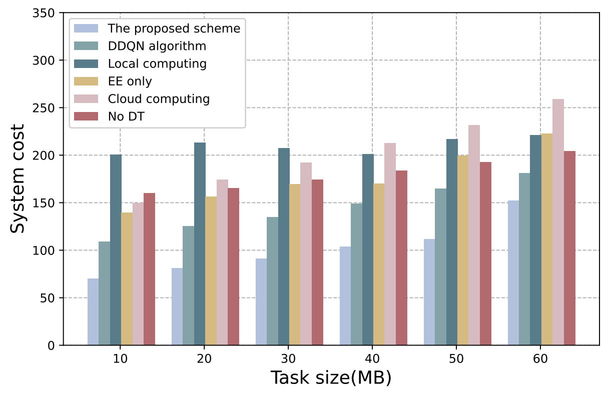

Fig. 5. Impact of task size on system cost. There are 20 UEs and 3 ESs

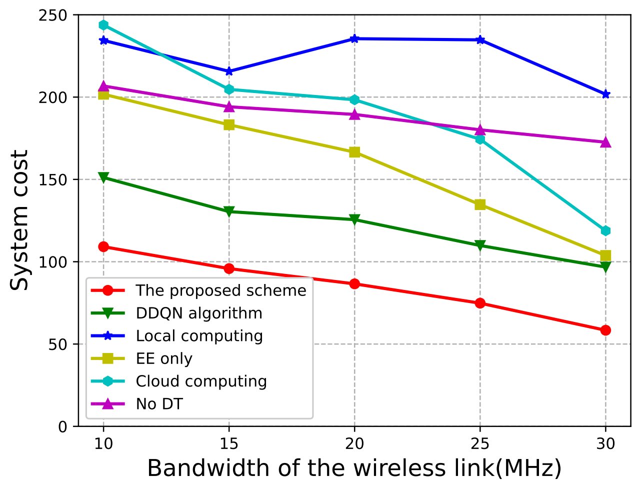

Fig. 6. Comparison of system cost with different bandwidths of each ES. There are 3 ESs.

The relationship between task size and system cost is depicted in Fig. 5. It is evident that the system cost of the other five benchmark schemes, excluding Local Computing, experiences a significant increase as the task size escalates. This can be attributed to the rise in both communication latency and energy consumption, along with a gradual surge in computational costs. In contrast, Local Computing, leveraging a mere 20 UEs for computation, does not necessitate data transmission. As the 'NO DT' scheme fails to harness global information, it fails to fully capitalize on the advantages of EEC, resulting in an average system cost that is 45.3%

higher when compared to our proposed scheme with DT. Our proposed scheme offers the flexibility to choose the offloading mode and achieves the lowest system cost through the joint optimization of computing and communication resources. In contrast, Cloud Computing has the highest system cost among all the schemes due to its excessive transmission delay.

Fig. 6 presents a comparison of the system cost under varying ES bandwidths, exhibiting a continuous decrease as the bandwidth increases. As outlined in (2) and (3), the bandwidth is directly related to the transmission rate of the wireless link, thereby leading to a substantial reduction in transmission delay. While centralized decision-based DDQN outputs encompass bandwidth allocation among different actions, attaining global optimality through centralized decision-making proves challenging. In contrast, the distributed decision-making approach employed in our proposed scheme, where each DT-UE makes decisions based on the current system state, enables effective bandwidth allocation and maximizes the utilization of limited communication resources. The proposed EEC collaboration scheme outperforms 'EE only' in terms of performance due to the abundant computational resources available in the CS. The 'EE only' approach falls short in computational delay due to the limited computational resources it relies on.

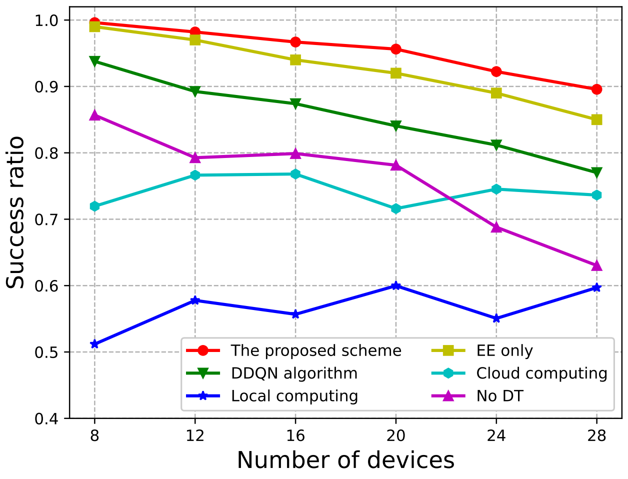

Fig. 7. Comparison of task success ratio with different numbers of UEs. There are 3 ESs with 20MHz bandwidth each.

Fig. 7 plots the task success ratio for each scheme at different numbers of UEs. The task success rate is defined as the ratio of the number of successful tasks to the total number of tasks, which measures the overall performance of the system in handling computational tasks. We can observe a decreasing trend in the task success rate as the number of UEs increases. This is primarily because the computational task size grows, leading to a higher demand for computational resources. However, the total bandwidth of the ES remains fixed, and since the UEs need to share limited communication resources among themselves, the communication rate decreases, ultimately resulting in task failures. In contrast, our proposed scheme effectively addresses this issue by utilizing the combined capabilities of end, edge, and cloud to split the task size. As a result, we are able to significantly increase the average task success rate by 43.9% compared to the benchmark schemes. It is worth noting that Local Computing and Cloud Computing exhibit some fluctuation. This is because the task completion rate for local computing depends entirely on the UE's computation and the size of the generated tasks, which is time-varying. For Cloud Computing, computational resources are sufficient but limited by the bandwidth of the wireless link.

C. Performance Evaluation of Deviation

As shown in Fig. 8, it is observed that the task success rate exhibits a decreasing trend as the task size increases. Specifically, when the task size is 10MB, both the 'EE only' and our proposed scheme show a similar task success ratio, with nearly all tasks being completed within the specified time. However, as the task size increases to 60MB, the 'EE only' becomes overwhelmed, resulting in a decline in the task success ratio to 82%. On the other hand, our proposed scheme efficiently divides the tasks and facilitates effective collaboration between the cloud and edge. This division of tasks allows us to make optimal use of computational resources, resulting in a task success ratio of over 90% even with the largest task size. It is worth noting that the absence of the DT significantly impacts the task success ratio, especially when the task size increases. The random division of tasks in 'No DT' fails to fully utilize the computational resources of the cloud. In contrast, our proposed scheme, with the assistance of DT, ensures a more efficient allocation of resources, thus maintaining a high success ratio even with larger task sizes.

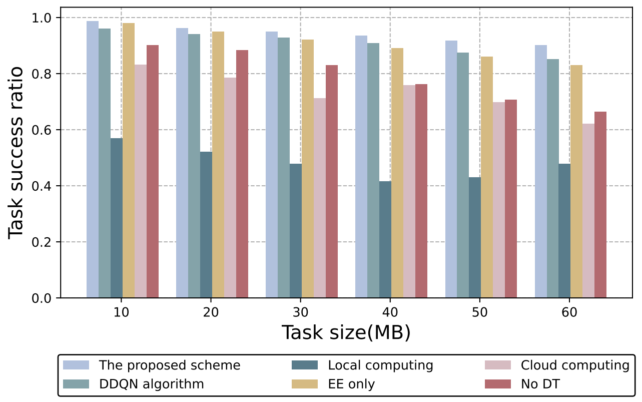

Fig. 8. Comparison of success ratio with different task size. There are 20 UEs and 3 ESs with 20M bandwidth each.

As shown in Fig. 9, we compare the variation of system cost for different bandwidths and deviations, which are present in all elements within the system, including CS, ESs, and UEs. When the deviations are constant, we observe that increasing the bandwidth of the ESs leads to a decrease in the corresponding system cost. This can be attributed to the fact that the UEs have obtained a larger bandwidth and selected ESs with better quality of service to offload tasks. Alternatively, when the ESs bandwidth remains constant, the system cost

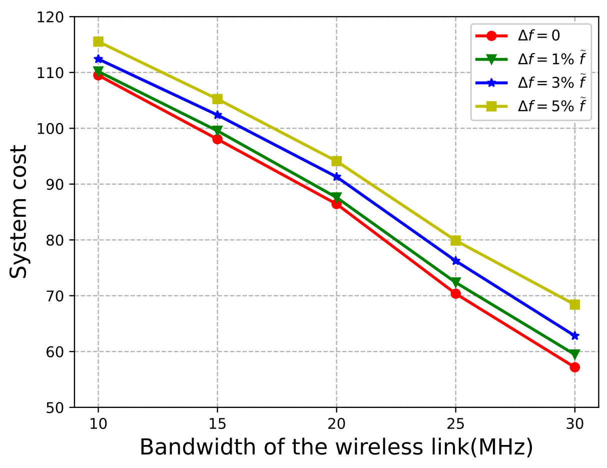

Fig. 9. Comparison of system cost with different deviations.

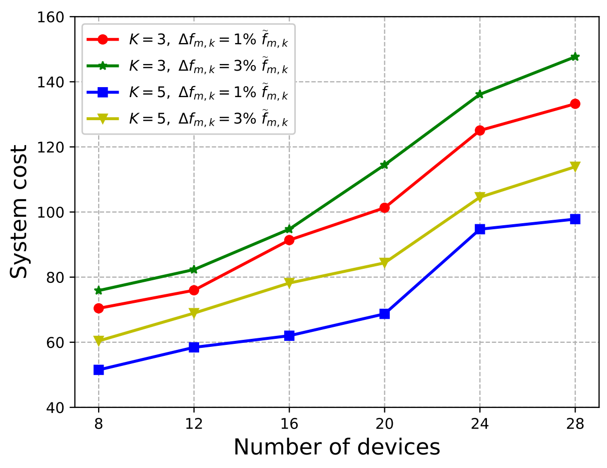

Fig. 11. Comparison of the system cost with the varying deviations and candidate ESs number K.

shows a positive correlation with the deviation. This can be explained by (11), (15), and (19), where a positive deviation implies that the estimated value is worse than the actual value, resulting in a larger actual delay compared to the estimated value.

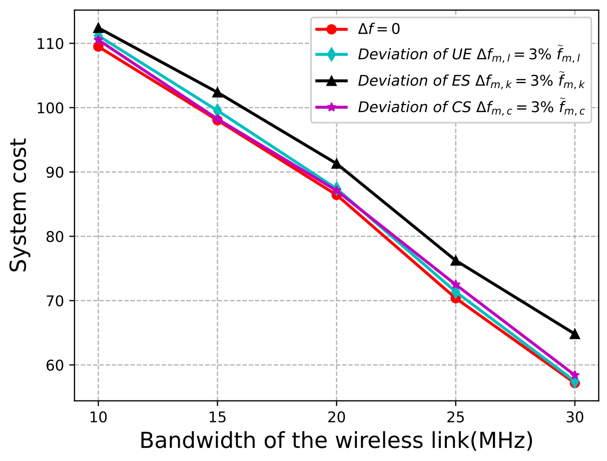

Fig. 10. Comparison of system cost with deviations of UEs, ESs, CS. Using our proposed scheme.

Fig. 10 illustrates the influence of deviations on the system cost under different bandwidth conditions. We consider scenarios with deviations for UEs, ESs, and CSs within our proposed EEC framework. The figure demonstrates that the system cost without deviations consistently outperforms other settings under the same bandwidth condition, while the deviation exhibits a positive correlation with the computation delay. On the other hand, CS possesses greater computational power compared to ESs, yet the impact of deviation is relatively smaller. The reason for this is that computational tasks are primarily executed in ESs to mitigate excessive latency.

As shown in Fig. 11, the variation of system cost with the number of UEs and available ESs K is depicted. It is evident from the figure that with a constant number of UEs, a higher number of available candidate ESs results in a reduction in system cost. This is attributed to the careful consideration of the dynamic characteristics of ESs during selection, as they are chosen from a larger candidate set. Furthermore, the figure also demonstrates an increase in the system cost as the number of UEs in the simulation area. This cost escalation is primarily due to the energy consumption associated with transmission and computation. The presence of deviations in ES introduces delays in task completion and increases the energy consumption. These factors contribute to inefficiencies in the system, which translate into costs incurred.

V. Conclusion

In this paper, we have investigated the problem of CCRA within the DT paradigm for EEC-based IIoT. Our study takes into account various factors relating to communication and computation, with particular emphasis on the deviation between real values and estimations. To tackle the highly dynamic and hybrid action space, we have proposed an algorithm based on DT-assisted MADDPG, aiming to minimize the overall system cost which includes latency and energy consumption. The algorithm optimizes the association policies, offloading mode, bandwidth allocation, and task split of UEs. To provide a comparison, we also present other benchmarking schemes separately. Extensive simulations have been conducted to analyze the impact of system parameters and deviations on the system cost. The obtained numerical results validate the effectiveness of our proposed solutions. Moving forward, we intend to further explore the modeling of DT in IIoT, including data transfer and instruction delivery.

References

[1] K. Wang, and W. Chen, "Energy-Efficient Communications in MIMO Systems Based on Adaptive Packets and Congestion Control With Delay Constraints," *IEEE Trans. Wireless Commun.*, vol. 14, no. 4, pp. 2169–2179, Apr. 2015.

[2] F. Wang, S. Cai and V. K. N. Lau, “Decentralized DNN Task Partitioning and Offloading Control in MEC Systems With Energy Harvesting Devices,” *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 173-188, Jan. 2023.

[3] D. Feng et al., “Toward ultrareliable low-latency communications: Typical scenarios, possible solutions, and open issues,” *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 94–102, Jun. 2019.

[4] K. B. Letaief, W. Chen, Y. Shi, J. Zhang and Y. -J. A. Zhang, “The Roadmap to 6G: AI Empowered Wireless Networks,” *IEEE Commun. Mag.*, vol. 57, no. 8, pp. 84-90, Aug. 2019.

[5] I. Tomkos, D. Klonidis, E. Pikasis and S. Theodoridis, “Toward the 6G Network Era: Opportunities and Challenges,” *IT Prof.*, vol. 22, no. 1, pp. 34-38, Jan.-Feb. 2020.

[6] B. Kar, W. Yahya, Y. -D. Lin and A. Ali, “Offloading Using Traditional Optimization and Machine Learning in Federated Cloud–Edge–Fog Systems: A Survey,” *IEEE Commun. Surv. Tut.*, vol. 25, no. 2, pp. 1199-1226, Apr.-Jun. 2023.

[7] Q. Luo, S. Hu, C. Li, G. Li and W. Shi, “Resource Scheduling in Edge Computing: A Survey,” *IEEE Commun. Surv. Tut.*, vol. 23, no. 4, pp. 2131-2165, Oct.-Dec. 2021.

[8] F. Wang and X. Zhang, “Joint Optimization for Traffic-Offloading and Resource-Allocation Over RF-Powered Backscatter Mobile Wireless Networks,” *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 5, pp. 1127-1142, Aug. 2021.

[9] K. Wang, D. Niyato, W. Chen and A. Nallanathan, “Task-Oriented Delay-Aware Multi-Tier Computing in Cell-Free Massive MIMO Systems,” *IEEE J. Sel. Areas. Commun.*, vol. 41, no. 7, pp. 2000-2012, Jul. 2023.

[10] S. Duan et al., “Distributed Artificial Intelligence Empowered by End-Edge-Cloud Computing: A Survey,” *IEEE Commun. Surv. Tut.*, vol. 25, no. 1, pp. 591-624, Jan.-Mar. 2023.

[11] A. Masaracchia, V. Sharma, B. Canberk, O. A. Dobre and T. Q. Duong , “Digital Twin for 6G: Taxonomy, Research Challenges, and the Road Ahead,” *IEEE Open J. Commun. Society.* , vol. 3, no. Nov, 2022 ,pp. 2137-2150.

[12] E. Ak, K. Duran, O. A. Dobre, T. Q. Duong and B. Canberk , “ T6CONF: Digital Twin Networking Framework for IPv6-Enabled Net-Zero Smart Cities ,” *IEEE Commun. Mag.*, vol. 61, no. 3, pp. 36-42, Mar. 2023.

[13] S. Mihai et al., “Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects,” *IEEE Commun. Surv. Tut.*, vol. 24, no. 4, pp. 2255-2291, Oct.-Dec. 2022.

[14] M. Fahim, V. Sharma, T.-V. Cao, B. Canberk, and T. Q. Duong , “ Machine Learning-based Digital Twin for Predictive Modeling in Wind Turbines,” *IEEE Access* , vol. 10, pp. 14184 - 14194 , Jan. 2022.

[15] Y. Li, J. A. Ansere, O. A. Dobre, and T. Q. Duong, ”Real-time Optimal Resource Allocation in Multiuser Mobile Edge Computing in Digital Twin Applications with Deep Reinforcement Learning,” in *Proc. IEEE Veh. Technol. Conf. Workshop*, Sep 2022, pp. 1-5.

[16] D. Van Huynh, V. -D. Nguyen, V. Sharma, O. A. Dobre and T. Q. Duong, “Digital Twin Empowered Ultra-Reliable and Low-Latency Communications-based Edge Networks in Industrial IoT Environment,” in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 5651-5656.

[17] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani and C. S. Hong, “Digital Twin of Wireless Systems: Overview, Taxonomy, Challenges, and Opportunities,” *IEEE Commun. Surv. Tut.*, vol. 24, no. 4, pp. 2230-2254, Oct.-Dec. 2022.

[18] Y. Yigit, B. Bal, A. Karameseoglu, T. Q. Duong and B. Canberk , “Digital Twin-Enabled Intelligent DDoS Detection Mechanism for Autonomous Core Networks,” *IEEE Commun. Standards Mag.* , vol. 6, no. 3, pp. 38-44, Sep. 2022.

[19] X. Dai et al., “Task Co-Offloading for D2D-Assisted Mobile Edge Computing in Industrial Internet of Things,” *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 480-490, Jan. 2023.

[20] S. Xia, Z. Yao, G. Wu and Y. Li, “Distributed Offloading for Cooperative Intelligent Transportation Under Heterogeneous Networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16701-16714, Sep. 2022.

[21] Y. Ding et al., “Online Edge Learning Offloading and Resource Management for UAV-Assisted MEC Secure Communications,” *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 54-65, Jan. 2023.

[22] K. Li, X. Wang, Q. He, B. Yi, A. Morichetta and M. Huang, “Cooperative Multiagent Deep Reinforcement Learning for Computation Offloading: A Mobile Network Operator Perspective,” *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24161-24173, Dec. 2022.

[23] R. Malik and M. Vu, “Energy-Efficient Joint Wireless Charging and Computation Offloading in MEC Systems,” *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 5, pp. 1110-1126, Aug. 2021.

[24] K. Wang, W. Chen, J. Li, Y. Yang and L. Hanzo, “Joint Task Offloading and Caching for Massive MIMO-Aided Multi-Tier Computing Networks,” *IEEE Trans. Commun.*, vol. 70, no. 3, pp. 1820-1833, Mar. 2022.

[25] D. Van Huynh, V. -D. Nguyen, S. Chatzinotas, S. R. Khosravirad, H. V. Poor and T. Q. Duong, “Joint Communication and Computation Offloading for Ultra-Reliable and Low-Latency With Multi-Tier Computing,” *IEEE J. Sel. Areas. Commun.*,vol. 41, no. 2, pp. 521-537, Feb. 2023.

[26] K. Wang et al., “Task Offloading With Multi-Tier Computing Resources in Next Generation Wireless Networks,” *IEEE J. Sel. Areas. Commun.*, vol. 41, no. 2, pp. 306-319, Feb. 2023.

[27] L. Chen, C. Shen, P. Zhou, and J. Xu, “Collaborative service placement for edge computing in dense small cell networks,” *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377-390, 1 Feb. 2021.

[28] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, “EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-Edge-Cloud orchestrated computing,” *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163-2176, Feb.15, 2021.

[29] Z. Zhou et al., ”Secure and Latency-Aware Digital Twin Assisted Resource Scheduling for 5G Edge Computing-Empowered Distribution Grids,” *IEEE Trans. Ind. Inform.*, vol. 18, no. 7, pp. 4933-4943, July 2022

[30] H. Liao et al., ”Cloud-Edge-Device Collaborative Reliable and Communication-Efficient Digital Twin for Low-Carbon Electrical Equipment Management,” *IEEE Trans. Ind. Inform.*, vol. 19, no. 2, pp. 1715-1724, Feb. 2023

[31] A. Hazra and T. Amgoth, “CeCO: Cost-Efficient Computation Offloading of IoT Applications in Green Industrial Fog Networks,” *IEEE Trans. Ind. Inform.*, vol. 18, no. 9, pp. 6255-6263, Sep. 2022.

[32] Z. Sun et al., “Cloud-Edge Collaboration in Industrial Internet of Things: A Joint Offloading Scheme Based on Resource Prediction,” *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17014-17025, Sep. 2022.

[33] Y. Ren, Y. Sun and M. Peng, “Deep Reinforcement Learning Based Computation Offloading in Fog Enabled Industrial Internet of Things,” *IEEE Trans. Ind. Inform.*, vol. 17, no. 7, pp. 4978-4987, Jul. 2021.

[34] H. Li, K. D. R. Assis, S. Yan and D. Simeonidou, “DRL-Based Long-Term Resource Planning for Task Offloading Policies in Multiserver Edge Computing Networks,” *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4151-4164, Dec. 2022.

[35] A. Suzuki and M. Kobayashi, “Multi-Agent Deep Reinforcement Learning for Cooperative Offloading in Cloud-Edge Computing,” in *Proc. IEEE Int. Conf. Commun.*, May 2022, pp. 3660-3666.

[36] D. Wei, N. Xi, X. Ma, M. Shojafar, S. Kumari and J. Ma, “Personalized Privacy-Aware Task Offloading for Edge-Cloud-Assisted Industrial Internet of Things in Automated Manufacturing,” *IEEE Trans. Ind. Inform.*, vol. 18, no. 11, pp. 7935-7945, Nov. 2022.

[37] L. U. Khan, W. Saad, D. Niyato, Z. Han and C. S. Hong, “Digital-Twin-Enabled 6G: Vision, Architectural Trends, and Future Directions,” *IEEE Commun. Mag.*, vol. 60, no. 1, pp. 74-80, Jan. 2022.

[38] T. Do-Duy, D. Van Huynh, O. A. Dobre, B. Canberk and T. Q. Duong, “Digital Twin-Aided Intelligent Offloading With Edge Selection in Mobile Edge Computing,” *IEEE Wireless Commun. Lett.*, vol. 11, no. 4, pp. 806-810, Apr. 2022.

[39] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre and T. Q. Duong, “Edge Intelligence-Based Ultra-Reliable and Low-Latency Communications for Digital Twin-Enabled Metaverse,” *IEEE Wireless Commun. Lett.*, vol. 11, no. 8, pp. 1733-1737, Aug. 2022.

[40] Q. Guo, F. Tang and N. Kato, “Federated Reinforcement Learning-Based Resource Allocation for D2D-Aided Digital Twin Edge Networks in 6G Industrial IoT,” *IEEE Trans. Ind. Inform.*, vol. 19, no. 5, pp. 7228-7236, May. 2023.

[41] W. Hou, H. Wen, H. Song, W. Lei and W. Zhang, “Multiagent Deep Reinforcement Learning for Task Offloading and Resource Allocation in Cybertwin-Based Networks,” *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16256-16268, Nov. 2021.

[42] W. Sun, H. Zhang, R. Wang and Y. Zhang, “Reducing Offloading Latency for Digital Twin Edge Networks in 6G,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12240-12251, Oct. 2020.

[43] X. Lin, J. Wu, J. Li, W. Yang and M. Guizani, “Stochastic Digital-Twin Service Demand With Edge Response: An Incentive-Based Congestion Control Approach,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2402-2416, 1 Apr. 2023.

[44] D. Van Huynh et al., “URLLC Edge Networks With Joint Optimal User Association, Task Offloading and Resource Allocation: A Digital Twin Approach,” *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7669-7682, Nov. 2022.

[45] X. Xu, Z. Liu, M. Bilal, S. Vimal and H. Song, "Computation Offloading and Service Caching for Intelligent Transportation Systems With Digital Twin," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20757-20772, Nov. 2022.

[46] X. Yuan, J. Chen, N. Zhang, J. Ni, F. R. Yu and V. C. M. Leung, "Digital Twin-Driven Vehicular Task Offloading and IRS Configuration in the Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 24290-24304, Dec. 2022.

[47] Y. Li, D. V. Huynh, T. Do-Duy, E. Garcia, T. Q. Duong , " Unmanned Aerial Vehicles-aided Edge Networks with Ultra-Reliable Low-Latency Communications: A Digital Twin Approach," *IET Signal Process.* , vol. 16, no.8, 2022, pp. 1-12.

[48] D. Van Huynh, V. -D. Nguyen, S. R. Khosravirad and T. Q. Duong, " Fairness-aware Latency Minimisation in Digital Twin-aided Edge Computing with Ultra-Reliable and Low-Latency Communications: A Distributed Optimisation Approach ," in *Proc. Asilomar Conf. Signals, Syst.Comput.* , Oct 2022, pp. 1045-1049.

[49] T. Q. Duong, D. Van Huynh, Y. Li, E. Garcia-Palacios and K. Sun, "Digital Twin-enabled 6G Aerial Edge Computing with Ultra-Reliable and Low-Latency Communications," in *Proc. Int. Conf. 6G Netw.*, Jul. 2022, pp. 1–5.

[50] H. Zhang, Y. Yang, B. Shang and P. Zhang, "Joint Resource Allocation and Multi-Part Collaborative Task Offloading in MEC Systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8877-8890, Aug. 2022.

[51] 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.

[52] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[53] R. Ding, Y. Xu, F. Gao, Shen, X. Shen, "Trajectory Design and Access Control for Air–Ground Coordinated Communications System With Multiagent Deep Reinforcement Learning," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5785-5798, Apr. 2022.

[54] X. Yang, H. Luo, Y. Sun and M. Guizani, "A Novel Hybrid-ARPPO Algorithm for Dynamic Computation Offloading in Edge Computing," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24065-24078, Dec. 2022.

Yuao Wang is currently working toward the Ph.D. degree with the Nanjing University of Posts and Telecommunications (NJUPT), JiangSu, China. His research interests include digital twin, edge intelligence, and resource optimization for end-edge-cloud systems.

Jingjing Fang received her bachelor's degree from the Civil Aviation University of China in 2023. Currently, she is working on her master's degree in the School of Communication and Information Engineering at Nanjing University of Posts and Telecommunications. Her current research interests include digital twins and mobile edge computing.

Yao Cheng received the B.S. degree from Xichang College, Xichang, China, in 2022. He is currently pursuing the M.S. degree with the School of Communication and Information Engineering, Nanjing University of Posts and Telecommunications, China. His current research interests include mobile edge caching, internet of vehicles, and deep reinforcement learning.

Hao She received the BS degree from the geographic information system, Nanjing University of Posts and Telecommunications, in 2019 and he is working toward the PhD degree at Nanjing University of Posts and Telecommunications now. His research interests include routing strategy, deep learning, and deep reinforcement learning.

Yongan Guo (IEEE Member) received the Ph.D. degrees from Nanjing University of Posts and Telecommunications (NJUPT) in 2018. He is currently an Professor in School of Communication and Information Engineering, NJUPT. And he is the founder of a novel research and development institution. His current research interests include Internet of Things, Edge Intelligence and future networks.

Gan Zheng (Fellow, IEEE) received the B.Eng. and M.Eng. degrees in electronic and information engineering from Tianjin University, Tianjin, China, in 2002 and 2004, respectively, and the Ph.D. degree in electrical and electronic engineering from The University of Hong Kong, Hong Kong, in 2008. He is currently a Professor of Connected Systems with the School of Engineering, University of Warwick, Coventry, U.K. His research interests include machine learning for wireless communications, UAV communications, mobile edge caching, full-duplex radio, and wireless power transfer. Prof. Zheng is the first recipient of the 2013 IEEE SIGNAL PROCESSING LETTERS Best Paper Award, and he also received the 2015 GLOBECOM Best Paper Award and the 2018 IEEE Technical Committee on Green Communications and Computing Best Paper Award. He was listed as a Highly Cited Researcher by Thomson Reuters/Clarivate Analytics in 2019. He currently serves as an Associate Editor for IEEE WIRELESS COMMUNICATIONS LETTERS.