# Digital Twin-Driven Collaborative Scheduling for Heterogeneous Task and Edge-End Resource via Multi-Agent Deep Reinforcement Learning

Chi Xu, *Senior Member, IEEE*, Zixuan Tang, Haibin Yu, *Senior Member, IEEE*, Peng Zeng, and Linghe Kong, *Senior Member, IEEE*

***Abstract*— With the interdisciplinary advances of mobile communication and edge computing, massive heterogeneous tasks are accessing wireless networks and competing for the edge-end computing and communication resources. Digital twin (DT), which establishes the digital models of physical objects for simulation, analysis and optimization, provides a promising method for network scheduling and management. This paper proposes a DT-driven edge-end collaborative scheduling algorithm for heterogeneous tasks and heterogeneous computing/communication resources. Specifically, multiple end devices (EDs) cooperate with each other to accomplish a complex job, where each ED can offload individual task to multiple edge servers (ESs) for parallel computing. By fully considering deadline requirements of heterogeneous tasks, maximum computing capabilities of ESs and EDs, computing resource estimation deviations of DT, maximum transmit powers of EDs and tolerable peak interference powers to coexisting EDs, we formulate a job completion time minimization problem to jointly optimize the edge-end task division, transmit power control, computing resource type matching and allocation. To solve this non-convex problem, we first reformulate it by multi-agent Markov decision process, where a compound reward leveraging latency reward and deadline reward according to the task criticality is designed. Then, we propose a multi-agent deep reinforcement learning-based scheduling algorithm, where Actor-Critic framework with estimation and target networks is designed for policy and value iterations. Meanwhile, a step-by-step $\epsilon$-greedy algorithm is proposed to balance exploration and exploitation, avoiding local optimal trap. Through offline centralized training by DT and online distributed execution by EDs, we realize edge-end collaborative computing for heterogeneous tasks. Experimental results demonstrate that, comparing with typical benchmark algorithms, the proposed algorithm converges with the highest reward and achieves the smallest job completion time, where the deadlines of heterogeneous tasks can be well satisfied respectively.**

***Index Terms*— Digital twin, collaborative scheduling, edge computing, task offloading, multi-agent deep reinforcement learning.**

Manuscript received 1 December 2022; revised 19 May 2023; accepted 4 August 2023. Date of publication 30 August 2023; date of current version 26 October 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1710900; in part by the National Natural Science Foundation of China under Grant 62173322 and Grant 92267108; in part by the Science and Technology Program of Liaoning Province under Grant 2023JH3/10200004 and Grant 2022JH25/10100005; and in part by the Youth Innovation Promotion Association, Chinese Academy of Sciences (CAS), under Grant 2019202. *(Corresponding authors: Haibin Yu; Linghe Kong.)*

Chi Xu and Peng Zeng are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110003, China, also with the Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China, and also with the Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China (e-mail: xuchi@sia.cn; zp@sia.cn).

Zixuan Tang and Haibin Yu are with the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110003, China, also with the Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China, also with the Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: tangzixuan@sia.cn; yhb@sia.cn).

Linghe Kong is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: linghe.kong@sjtu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3310066.

Digital Object Identifier 10.1109/JSAC.2023.3310066

## I. Introduction

With the rapid development of 5G, more and more devices are accessing the Internet and interconnecting humans, machines, and things with each other, towards Internet of Everything [1]. Thus, there is an explosion that massive heterogeneous tasks are delivering over the 5G network. The heterogeneous tasks can be video/media tasks that require broadband communications, sensing/measuring tasks that require low-power communications, and industrial control tasks that require realtime computing and deterministic communications. To accomplish a complex job, we need to coordinate these heterogeneous tasks. For example, when an engineer teleoperates a robot for high precision machining, the heterogeneous tasks include holographic media and force-feedback control data for human's visual-haptic-auditory perception, and multi-sensor multi-controller data for robot's positioning, teaching and learning [2]. When these heterogeneous tasks implement high-concurrent access [3], they must compete for the limited communication resources distributed in temporal, spatial and frequency domains, such as timeslot, power, antenna, channel, and subcarrier. This will cause communication conflicts, which certainly decrease the quality of experience (QoE).

To enhance the QoE, multi-access edge computing (MEC) is proposed to process tasks nearby the end devices (EDs) and reduce the task processing latency. For example, by deploying edge server (ES) at the base station (BS), BS can implement some network management functions and provide computing resources for task processing. Thus, MEC-enhanced 5G is currently regarded as a key enabler for vertical industries. However, employing MEC will also introduce new problems. First, task offloading to ESs will consume the heterogeneous communication resources, which certainly exacerbates

the communication resource competition problem. Second, the computing resources distributed at EDs and ESs are also heterogeneous, wherein the computing resources can be supplied by CPU, GPU, or others. In this way, the edge-end computing capabilities for different tasks are also different. Thus, there remains a challenge that how to schedule the heterogeneous computing and communication resources for the massive heterogeneous tasks to realize edge-end collaboration.

Digital twin (DT), which establishes the digital models of physical objects for simulation, analysis and optimization, provides a novel way to address the above challenges. DT is initially proposed for cyber-physical production systems to achieve smart manufacturing in Industry 4.0 [4]. Since the proposal of DT, it arises great interests from both academia and industries. With the interdisciplinary advances in 5G, cloud/edge computing, big data, and artificial intelligence, the capability of DT is continuously enhanced, empowering not only one-way information mirroring and simulations, but also round-trip interaction and operations. Thus, DT is quickly diffusing in numerous different industries, such as smart city, Internet of vehicles, and 5G. Currently, DT is regarded as a key technology enabling 6G [5]. In particular, cybertwin [6] and networked twin [7] are proposed and investigated for network management and automation. Furthermore, DT network [8], and DT edge network [9] are formulated.

With DT, the heterogeneous computing and communication resources can be virtualized and modelled for flexible scheduling. Motivated by this, we employ DT to collaboratively schedule the edge-end heterogeneous computing and communication resources for the heterogeneous tasks with different deadline requirements. Specifically, we consider a multi-ES multi-ED scenario with a synchronous DT deployed at the cloud server (CS). To accomplish a complex job, EDs cooperate with each other and offload individual tasks to multiple ESs for parallel computing via the scheduling of DT, where different tasks require different types of computing resources and have different task deadlines. To minimize the job completion time (JCT), we employ multi-agent deep reinforcement learning (MADRL) and propose MADRL-based heterogeneous task and resource collaborative scheduling (MADRL-HTRCS) algorithm.

The main contributions of this paper are summarized as follows.

1) We study a general scenario with single-CS, multi-ES and multi-ED, where the computing resources of ESs and EDs are heterogeneous and can support different kinds of tasks. We utilize DT to virtualize and model the heterogeneous computing resources, during which DT's estimation deviations between actual and estimated computing resources for ESs and EDs are considered. Meanwhile, the tasks are also heterogeneous and can be totally/partially/none offloaded to multiple ESs for parallel computing. That is to say, each task is processed by the cooperation between ED and multiple ESs.
2) By fully considering the deadline requirements of heterogeneous tasks, the maximum computing capabilities of both EDs and ESs, the computing resource estimation deviations of DT, the maximum transmit powers of EDs and the peak interference powers to coexisting EDs, we establish a job completion time minimization (JCTM) problem to optimize the edge-end task division, transmit power control, computing resource type matching and allocation. Due to the non-convexity of the JCTM problem, we reformulate it by multi-agent Markov decision process (MDP), where each ED is modelled as an agent interacting with environment and other agents independently. Furthermore, we design a compound reward leveraging latency reward and deadline reward according to the task criticality.
3) To approximate an optimal solution, we propose the MADRL-HTRCS algorithm that supports offline centralized training by DT and online distributed execution of EDs. Specifically, we employ the Actor-Critic (AC) framework and design estimation and target AC networks for policy and value iterations. Moreover, a step-by-step $\epsilon$-greedy algorithm is applied to balance exploration and exploitation. Extensive experiments validate the effectiveness and superiority of the proposed algorithm by comparing with some benchmark algorithms.

The rest of this paper is organized as follows. In Section II, the recent works on task-resource scheduling are reviewed. In Section III, the DT-based system model is presented. After that, we establish the JCTM problem and reformulate it by multi-agent MDP in Section IV. Then, we propose the MADRL-HTRCS algorithm in Section V, and validate its effectiveness by extensive experiments in Section VI. Finally, the whole work is concluded in Section VII.

## II. Related Work

Task-resource collaborative scheduling is the basis of cloud/edge computing and has attracted broad interests during the past decade. Previous works have comprehensively investigated different MEC scenarios, employed different theories or algorithms, and optimized different objectives to achieve different goals. Specifically, the MEC scenario is related to the numbers of ESs and EDs, including single-ES multi-ED, multi-ES single ED, multi-ES multi-ED and so on. Herein, a task can be divided or not, and offloaded to one ES or multiple ESs. Typical task offloading schemes include binary offloading and partial offloading [10]. Meanwhile, the scheduling objectives can be tasks (e.g., offloading ratio), computing resources (e.g., CPU cycle) and communication resources (e.g., transmit power, bandwidth, channel, subcarrier). On this basis, existing works formulate different optimization problems aiming at minimizing energy, latency or cost, and propose different algorithms based on convex optimization, machine learning, etc. In the following, we summarize the most related works from the perspective of optimization objectives.

### A. Energy Consumption Minimization

Energy consumption is always a key indicator for wireless networks, especially for energy-constrained EDs. The energy

consumption mainly includes the communication energy consumption for task offloading and the computing energy consumption for task processing.

For single-CS single-ES multi-ED scenario, [11] employs deep Q-network (DQN) to study the long-term energy consumption minimization problem under the constraint of computing resources and latency. For multi-ES single-ED scenario, to minimize the overall energy consumption while satisfying the latency limit, [12] investigates the joint optimization of multi-task offloading, non-orthogonal multiple access (NOMA) transmission, and computing resource allocation. Similarly, [13] establishes a three-layer offloading framework and investigates the overall energy consumption minimization problem subject to latency constraints from EDs. Moreover, [14] investigates stochastic computation offloading and resource allocation problem to optimize long-term energy efficiency using Lyapunov optimization and asynchronous AC algorithm in DT network. Reference [15] proposes a deep learning-based user association and resource allocation algorithm which is trained by DT to minimize the maximum normalized energy consumption.

### B. Latency Minimization

Although MEC can help process complex tasks for computation-intensive EDs, it may also introduce traffic conflicts and increase communication latency. Thus, minimizing the latency, which consists of communication latency and computing latency, is also very important, especially for time-sensitive tasks.

For single-ES multi-ED scenario, [16] investigates the long-term caching placement and resource allocation problem, and adopts deep reinforcement learning (DRL) to minimize the content delivery latency. In contrast, [17] studies multi-ES single-ED scenario and optimizes the NOMA-based transmission duration and task division to multiple ESs. Furthermore, [18] employs convex optimization and MADRL to jointly optimize sub-channel assignment, offloading decision, and computing resource allocation in multi-ES multi-ED scenario. Reference [19] proposes a risk-sensitive DRL algorithm to minimize the offloading and computing latency of all tasks constrained by given energy capacity. By modelling the user mobility and environment dynamics in DT, [20] proposes an AC-based DRL algorithm to minimize the offloading latency under the constraint of service migration cost for user mobility. Moreover, with DT and blockchain, [21] minimizes the latency for edge association by federated MADRL.

### C. Cost Minimization

In addition to optimizing energy consumption and latency respectively, more recent works focus on minimizing the system cost, which is usually defined as a weighted sum of energy consumption and latency. In this way, the system performance can be optimized according to the requirements of specific tasks.

For single-ES multi-ED scenario, [22] combines AC and DQN algorithms to jointly optimize the task offloading policy and channel allocation for time-varying channels. Reference [23] considers the case that multiple EDs offload their tasks via NOMA to multiple ESs, and employs reinforcement learning and matching game theory to solve the joint task scheduling and resource allocation problem with respect to task, power, subcarrier, and computing frequency. Furthermore, [24] exploits MADRL to optimize offloading decisions and transmit powers for edge-end orchestrated resource allocation of industrial wireless networks. Based on asynchronous advantage AC and DQN algorithms, [25] optimizes offloading decisions, node selection, bandwidth and computing resource allocations for single-CS multi-ES multi-ED scenario, wherein DT is utilized in the cloud. More recently, [26] considers multi-ES single-ED scenario based on DT and blockchain, and proposes a decision tree and double DQN (DDQN) solution for intelligent task offloading. Moreover, [27] proposes adaptive DT for vehicular edge network and employs MADRL to minimize the offloading cost.

Besides the above works, some works also define special optimization objectives for task-resource scheduling. For example, [28] formulates a multi-objective problem to minimize latency and energy consumption simultaneously, and employs MADRL to make an optimal offloading decision for cloud-edge-end computing. Reference [29] proposes an end-to-end DRL algorithm to maximize the number of tasks before their respective deadlines and minimize energy consumption simultaneously. Reference [10] formulates a computing rate maximization problem subject to the long-term data queue stability and average power constraints, and employs Lyapunov and DRL to achieve the optimal computing performance. In addition, by proposing a D3PG-based task offloading algorithm, [30] tries to maximize QoE with respect to service latency, energy consumption and task success rate. Reference [31] employs DQN to maximize average QoE for DT-empowered Internet of vehicles.

From the aforementioned works, we can observe that existing studies on task-resource collaborative scheduling by DT are still on the early stage. More importantly, few existing works consider the heterogeneous computing resources problem, where different tasks require different types of computing resources. This motivates us to investigate the DT-driven edge-end heterogeneous computing and communication collaborative scheduling for heterogeneous tasks.

## III. System Model

In this section, we present the system model, including the network model, communication model, edge and local computing models. For ease of reading, we list the key notations in Table I.

### A. Digital Twin-Based Network Model

In this paper, we consider a general single-CS, multi-ES and multi-ED scenario. As shown in Fig. 1, there are one DT-embedded CS, $N$ ES-enhanced BSs and $M$ resource-constrained EDs in the physical space. Specifically, with full consideration of the strong computing capability and multi-type computing resources of CS, DT is deployed in CS

TABLE I
SUMMARY OF KEY NOTATIONS

| Notation | Definition | Notation | Definition |
|---|---|---|---|
| $N$ | Number of ESs | $D_m$ | Task's data size of $m$-ED |
| $M$ | Number of EDs | $C_m$ | Required CPU cycles for $m$-ED |
| $F_{\max,n}$ | Maximum computing resource of $n$-ES | $u_{m,n}$ | Resource type matching decision between $m$-ED and $n$-ES |
| $F_{\max,m}$ | Maximum computing resource of $m$-ED | $v_{m,n}$ | Task division ratio of $m$-ED for $n$-ES |
| $f_{m,n}$ | DT's estimated computing resource of $n$-ES for $m$-ED | $\tilde{T}^{\text{Comp}}_{m,n}$ | Estimated computing latency of $m$-ED by $n$-ES |
| $\Delta f_{m,n}$ | DT's estimation deviation of $n$-ES for $m$-ED | $\tilde{T}^{\text{Comp}}_{m}$ | Estimated computing latency of $m$-ED |
| $f_m$ | DT's estimated computing resource of $m$-ED | $\Delta T^{\text{Comp}}_{m,n}$ | Computing latency deviation of $m$-ED by $n$-ES |
| $\Delta f_m$ | DT's estimation deviation of $m$-ED | $\Delta T^{\text{Comp}}_{m}$ | Computing latency deviation of $m$-ED |
| $W_{m,n}$ | Channel bandwidth between $m$-ED and $n$-ES | $T^{\text{Comm}}_{m,n}$ | Communication latency from $m$-ED to $n$-ES |
| $\sigma_n^2$ | Noise power at $n$-ES | $T^{\text{Comp}}_{m,n}$ | Actual computing latency of $m$-ED by $n$-ES |
| $p_m$ | Transmit power of $m$-ED | $T^{\text{Edge}}_{m,n}$ | Edge computing latency of $m$-ED by $n$-ES |
| $P_{\max}$ | Maximum transmit power of ED | $T^{\text{Comp}}_{m}$ | Actual computing latency of $m$-ED |
| $I_p$ | Peak interference power of ED | $T^{\text{Local}}_{m}$ | Local computing latency of $m$-ED |
| $g_{m,n}$ | Channel power gain from $m$-ED to $n$-ES | $T^{\text{Edge}}_{m}$ | Edge computing latency of $m$-ED by multiple ESs |
| $g_{m,m'}$ | Channel power gain from $m$-ED to $m'$-ED | $T_m$ | Task processing latency of $m$-ED |
| $R_{m,n}$ | Communication rate between $m$-ED and $n$-ES | $T_{\max,m}$ | Task deadline of $m$-ED |

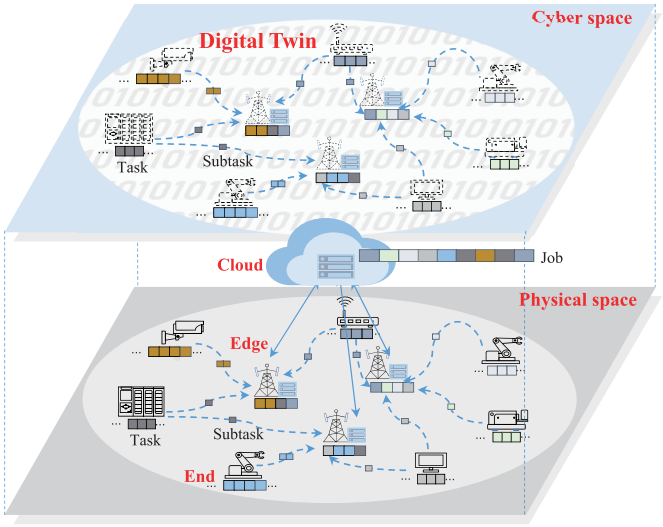



Fig. 1. System model.

to mirror and model all physical network elements into the cyber space. In this way, DT senses the heterogeneous tasks, measures the heterogeneous computing and communication resources, trains the following proposed scheduling algorithm and schedules the heterogeneous tasks and resources.

To accomplish a complex job, multiple EDs cooperate with each other, where each ED implements individual task, respectively. The tasks are heterogeneous that have different data sizes, require different types of computing resources and should be completed before different deadlines. For $m$-ED, the data size and task deadline are denoted as $D_m$ and $T_{\max,m}$ $(m = 1, \ldots, M)$. Due to the limited computing resource, an ED can divide its task into multiple subtasks and offload them to different ESs for parallel computing. The task may be totally offloaded for full edge computing, none offloaded for full local computing or partially offloaded. Specifically, $m$-ED offloads $v_{m,n}D_m$ task to $n$-ES $(n = 1, \ldots, N)$, where $v_{m,n} \in [0, 1]$ is the task division ratio of $m$-ED for $n$-ES. When $v_{m,n} = 0$, $m$-ED does not offload any task to $n$-ES. On the contrary, $m$-ED offloads the total task to $n$-ES when $v_{m,n} = 1$.

To guarantee all subtasks are processed for the task of $m$-ED, we have the task division constraint as

$$\sum_{n=0}^{N} v_{m,n} = 1, \tag{1}$$

where $n = 0$ indicates the subtask for local processing.

### B. Communication Model

According to the scheduling by DT, $m$-ED employs the transmit power $p_m$ for task offloading. The transmit power of $m$-ED must be constrained by its hardware capability, i.e.,

$$0 \le p_m \le P_{\max}, m = 1, \ldots M, \tag{2}$$

where the maximum transmit powers of all EDs are assumed the same as $P_{\max}$.

Meanwhile, the task offloading of EDs on the same wireless channel may cause co-channel interference with each other. Each ED has an interference temperature which is the peak interference power that an ED can tolerate. For simplicity, we assume the tolerable peak interference powers of EDs are the same as $I_p$. When $m$-ED performs task offloading, its transmit power is constrained by all coexisting EDs, where $m^*$-ED with the maximum channel power gain from $m$-ED imposes the strongest constraint. That is $m^* = \arg\max g_{m,m'}, (m' = 1, \ldots, M, m' \neq m)$. Then, by fully considering all possible offloading interferences to $m^*$-ED, the transmit power of $m$-ED is constrained by $p_m g_{m,m^*} + \sum_{m'=1,m'\neq m}^{M} p_{m'} g_{m',m^*} \le I_p$. That is,

$$p_m \le \frac{I_p - \sum\limits_{m'=1,m'\neq m}^{M} p_{m'} g_{m',m^*}}{g_{m,m^*}}, \tag{3}$$

where $g_{m,m^*}$ and $g_{m',m^*}$ denote the channel power gains from $m$-ED and $m'$-ED to $m^*$-ED, respectively. Herein, the channel between any pair of EDs and/or ESs is assumed to be

symmetric and the channel state information can be accurately evaluated by DT for modelling and scheduling. Note that this assumption can be easily extended to the asymmetric channels with/without evaluation error.

By transmit power control for wireless communication, we can calculate the task offloading rate between $m$-ED and $n$-ES as

$$R_{m,n} = W_{m,n} \log_2 \left( 1 + \frac{p_m g_{m,n}}{\sum\limits_{m'=1, m' \neq m}^{M} p_{m'} g_{m',n} + \sigma_n^2} \right), \tag{4}$$

where $W_{m,n}$ denotes the bandwidth between $m$-ED and $n$-ES for task offloading, $\sigma_n^2$ denotes the noise at $n$-ES, $g_{m,n}$ and $g_{m',n}$ denote the channel power gains from $m$-ED and $m'$-ED to $n$-ES, respectively.

According to equation (4), we can further calculate the communication latency for task offloading.

$$T_{m,n}^{\text{Comm}} = \frac{v_{m,n} D_m}{R_{m,n}}. \tag{5}$$

It is observed that equations (3) and (4) take into account the interferences of all possible EDs that offload tasks to ESs. When the number of EDs is large, the co-channel interferences become large, which limits the transmit powers of EDs and certainly reduces their offloading rates. That is to say, when EDs perform high-concurrent task offloading, there will be significant computing and communication resources competitions among EDs. In this way, an efficient task-resource collaborative scheduling algorithm is critical for job completion.

### C. Edge Computing Model

The computing resources of ESs or EDs are single-type and heterogeneous, e.g., CPU, GPU. For example, an ED processing image or media task only equips with GPU, while an ED processing sensing or control data only equips with CPU. Obviously, a GPU-type task offloaded to a CPU-equipped ES will not be processed with high efficiency. Thus, we assume that an ES with given type of computing resource, can only process the task offloaded by the ED with the same type of computing resource.

By mapping the physical computing resources of EDs and ESs at DT, DT can match the types of heterogeneous computing resources distributed at ESs and EDs. The heterogeneous computing resource type matching decision is expressed as

$$u_{m,n} = \begin{cases} 1, & \text{if } o_n \otimes o_m = 0, \\ 0, & \text{if } o_n \otimes o_m = 1, \end{cases} \tag{6}$$

where $o_n$ and $o_m$ indicate the computing resource types of $n$-ES and $m$-ED, respectively. $\otimes$ is the exclusive OR operation. $u_{m,n} = 1$ indicates that the computing resource types of $m$-ED and $n$-ES are the same. Otherwise, the computing resource types are different and $n$-ES cannot support the task processing for $m$-ED.

By matching the heterogeneous computing resources among ESs and EDs, DT further evaluates the edge-end resources and schedules tasks for parallel computing. The computing resource is measured by computing rate $f_{m,n}$, namely the number of computing cycles per second. When $m$-ED offloads $v_{m,n} D_m$ task to $n$-ES, the edge computing latency estimated by DT is calculated as

$$\tilde{T}_{m,n}^{\text{Comp}} = \frac{v_{m,n} D_m C_m}{f_{m,n}}, \tag{7}$$

where $C_m$ is the required cycles for computing 1 Byte task.

However, DT may have a computing resource estimation deviation $\Delta f_{m,n}$ which may be either positive or negative. In this way, the actual computing resource of $n$-ES allocated to $m$-ED is calculated as $f_{m,n} + \Delta f_{m,n}$ which should satisfy

$$0 \leq f_{m,n} + \Delta f_{m,n} \leq F_{\max,n}, \tag{8}$$

and

$$\sum_{m=1}^{M} u_{m,n} (f_{m,n} + \Delta f_{m,n}) \leqslant F_{\max,n}, \tag{9}$$

where $F_{\max,n}$ denotes the maximum computing rate of $n$-ES.

Then, we can calculate the computing latency deviation between the actual value and the estimated value, i.e.,

$$\Delta T_{m,n}^{\text{Comp}} = -\frac{v_{m,n} D_m C_m \Delta f_{m,n}}{f_{m,n} (f_{m,n} + \Delta f_{m,n})}. \tag{10}$$

For the subtasks offloaded from $m$-ED, the actual computing latency by $n$-ES is calculated as

$$T_{m,n}^{\text{Comp}} = \tilde{T}_{m,n}^{\text{Comp}} + \Delta T_{m,n}^{\text{Comp}}. \tag{11}$$

Furthermore, the edge computing latency for the subtasks of $m$-ED by $n$-ES is calculated as

$$T_{m,n}^{\text{Edge}} = T_{m,n}^{\text{Comm}} + T_{m,n}^{\text{Comp}}, \tag{12}$$

wherein the computing results' feedback latency from $n$-ES to $m$-ED is ignored since the data size of feedback is generally very small and can be carried back by the acknowledged information during communication.

As the task of $m$-ED is divided into multiple subtasks and offloaded to multiple ESs for parallel processing, the edge computing latency for the total task of $m$-ED is calculated as

$$T_m^{\text{Edge}} = \max_{n=1,\ldots,N} \{ u_{m,n} T_{m,n}^{\text{Edge}} \}. \tag{13}$$

### D. Local Computing Model

Similar to the edge computing model, the local computing resource of $m$-ED estimated by DT is denoted as $f_m$. Then, the local computing latency estimated by DT is calculated as

$$\tilde{T}_m^{\text{Comp}} = \frac{v_{m,0} D_m C_m}{f_m}. \tag{14}$$

There is also estimation deviation $\Delta f_m$ which can be obtained by DT in advance [20], [26]. Thus, we have

$$f_m = F_{\max,m} - \Delta f_m, \tag{15}$$

where $F_{\max,m}$ is the maximum computing rate of $m$-ED decided by the physical hardware. This is because each ED should utilize the full computing resource to process the

local task for latency reduction. The local computing latency deviation is calculated as

$$\Delta T_m^{\text{Comp}} = -\frac{v_{m,0} D_m C_m \Delta f_m}{f_m F_{\max,m}}. \tag{16}$$

In this way, the actual local computing latency by $m$-ED is calculated as

$$T_m^{\text{Local}} = \tilde{T}_m^{\text{Comp}} + \Delta T_m^{\text{Comp}}. \tag{17}$$

## IV. Problem Formulation and Transformation

### A. Job Completion Time Minimization Problem

As a task is completed by DT-driven edge-end collaborative computing, the task processing latency of $m$-ED is calculated as the maximum latency for edge computing and local computing, i.e.,

$$T_m = \max\left(T_m^{\text{Edge}}, T_m^{\text{Local}}\right), \tag{18}$$

which includes the cases of none, partial and total offloading.

Then, we can calculate JCT as $\sum_{m=1}^{M} T_m$, where a job is completed by the sequel completion of all tasks. Furthermore, with full consideration of heterogeneous tasks' requirements, heterogeneous computing and communication resources constraints, the JCTM problem is formulated as

$$\text{JCTM}: \min_{\mathcal{U},\mathcal{V},\mathcal{P},\mathcal{F}} \quad \sum_{m=1}^{M} T_m, \tag{19}$$

$$\text{s.t.} \quad (1),(2),(3),(6),(8),(9),$$
$$T_m \le T_{\max,m}, \tag{20}$$

where $\mathcal{U} = \{u_{m,n}\}_{M\times N}$, $\mathcal{V} = \{v_{m,n}\}_{M\times(N+1)}$, $\mathcal{P} = \{p_m\}_M$, and $\mathcal{F} = \{f_{m,n}\}_{M\times N}$ are the computing resource type matching decisions, task division ratios, transmit powers of EDs, and computing resources of ESs.

In the JCTM problem, we consider the task division constraint as (1), the transmit power constraints as (2) and (3), the computing resource type matching decision as (6), the computing capability constraints as (8) and (9), and the task deadline constraint as (20). Obviously, there are both integer and real variables, which are coupled with each other in the JCTM problem. Thus, it is a mixed integer non-linear programming problem, which is NP-hard and cannot be solved within a polynomial time by common methods such as convex optimization [32]. Thus, we employ multi-agent MDP to transform the problem for MADRL solution.

### B. Problem Transformation by Multi-Agent MDP

By the estimation and scheduling of DT in cloud, EDs and ESs cooperate with each other to complete a complex job. In this way, any action of an ED may influence the total system state such as co-channel interference, task division, and resource scheduling by DT. Meanwhile, the state transformation is also related with previous state and action. Thus, we employ multi-agent MDP to reformulate the JCTM problem. The multi-agent MDP is described by five tuples $\langle \mathcal{M}, \mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{R} \rangle$, where $\mathcal{M}$, $\mathcal{S}$, $\mathcal{A}$, $\mathcal{Z}$, and $\mathcal{R}$ denote the agent set, state space, action space, state transition probability, and reward function, respectively.

*1) Agent Set $\boldsymbol{\mathcal{M}}$:* Aiming at minimizing JCT, each ED acts as an agent to learn its computing resource type matching decision, task division ratio, transmit power, and computing resource allocation. Thus, $M$ EDs form an agent set $\mathcal{M} = \{1, \ldots, M\}$.

*2) State Space $\boldsymbol{\mathcal{S}}$:* The state space describes the running status of tasks as well as edge-end computing and communication resources, which can be observed by agent and evaluated by DT. At each decision epoch $t$, the state $\boldsymbol{s}_m(t)$ of $m$-agent is characterized by data size, computing resource requirement, task deadline, computing resource estimation deviation, bandwidth and channel power gain, i.e.,

$$\boldsymbol{s}_m(t) = \{D_m(t), C_m(t), T_{\max,m}(t), \Delta f_m(t), \boldsymbol{\Delta} F_m^{\text{Edge}}(t), \boldsymbol{W}_m(t), \boldsymbol{G}_m(t)\}, \tag{21}$$

where $\boldsymbol{\Delta} F_m^{\text{Edge}}(t) = \{\Delta f_{m,n}(t)\}_{1\times N}$, $\boldsymbol{W}_m(t) = \{W_{m,n}(t)\}_{1\times N}$ and $\boldsymbol{G}_m(t) = \{\{g_{m,n}(t)\}_{1\times N}, \{g_{m,m'}(t)\}_{1\times M}\}$. Furthermore, we define the total state space of all agents at the decision epoch $t$ as $\boldsymbol{s}(t) = \{\boldsymbol{s}_m(t)\}_M$.

*3) Action Space $\boldsymbol{\mathcal{A}}$:* The action space presents the policies of all agents. At each decision epoch $t$, $m$-agent performs action $\boldsymbol{a}_m(t)$ according to the whole state $\boldsymbol{s}(t)$ subject to the constraints in the JCTM problem. The action describing the computing resource type matching decision, task division ratio, transmit power of ED, and computing resource allocation of ES, is given by

$$\boldsymbol{a}_m(t) = \{\boldsymbol{u}_m(t), \boldsymbol{v}_m(t), p_m(t), \boldsymbol{f}_m(t)\}, \tag{22}$$

where $\boldsymbol{u}_m(t) = \{u_{m,n}(t)\}_{1\times N}$, $\boldsymbol{v}_m(t) = \{v_{m,n}(t)\}_{1\times(N+1)}$, and $\boldsymbol{f}_m(t) = \{f_{m,n}(t)\}_{1\times N}$. Furthermore, we define the total action space of all agents at the decision epoch $t$ as $\boldsymbol{a}(t) = \{\boldsymbol{a}_m(t)\}_M$.

*4) State Transition Probability $\boldsymbol{\mathcal{Z}}$:* At each decision epoch $t$, the state transition probability $z_m(t)$ describes the probability that $\boldsymbol{s}_m(t)$ transfers to $\boldsymbol{s}_m(t+1)$ when $m$-agent performs action $\boldsymbol{a}_m(t)$, namely $z_m\left(\boldsymbol{s}_m(t+1); \boldsymbol{s}_m(t), \boldsymbol{a}_m(t)\right)$.

*5) Reward Function $\boldsymbol{\mathcal{R}}$:* The reward presents the award or penalty for agent when it takes action at a given state. For multi-agent MDP, $M$ agents interact with environment and cooperate with each other according to the state and the policy to obtain individual reward $r_m(t)$. Specifically, at each decision epoch $t$, $m$-agent performs action $\boldsymbol{a}_m(t)$ at state $\boldsymbol{s}_m(t)$, obtains the reward $r_m(t)$ and moves to the next state $\boldsymbol{s}_m(t+1)$.

Fully considering the objective and constraints in the JCTM problem, we design the reward as the sum of latency reward and deadline reward. The latency reward is defined as $r_m^{\text{Latency}}(t) = -T_m(t)$, while the deadline reward is defined as $r_m^{\text{DDL}}(t) = T_{\max,m}(t) - T_m(t)$. In this way, when the latency exceeds the deadline, there is a negative reward, namely penalty.

As there are diverse requirements of heterogeneous tasks, we design the compound reward of $m$-agent as

$$r_m(t) = r_m^{\text{Latency}}(t) + \rho_m r_m^{\text{DDL}}(t), \tag{23}$$

where $\rho_m$ is the weight parameter set according to the deadline requirements of heterogeneous tasks. That is, the larger value of the weight parameter, the stricter deadline of this task.

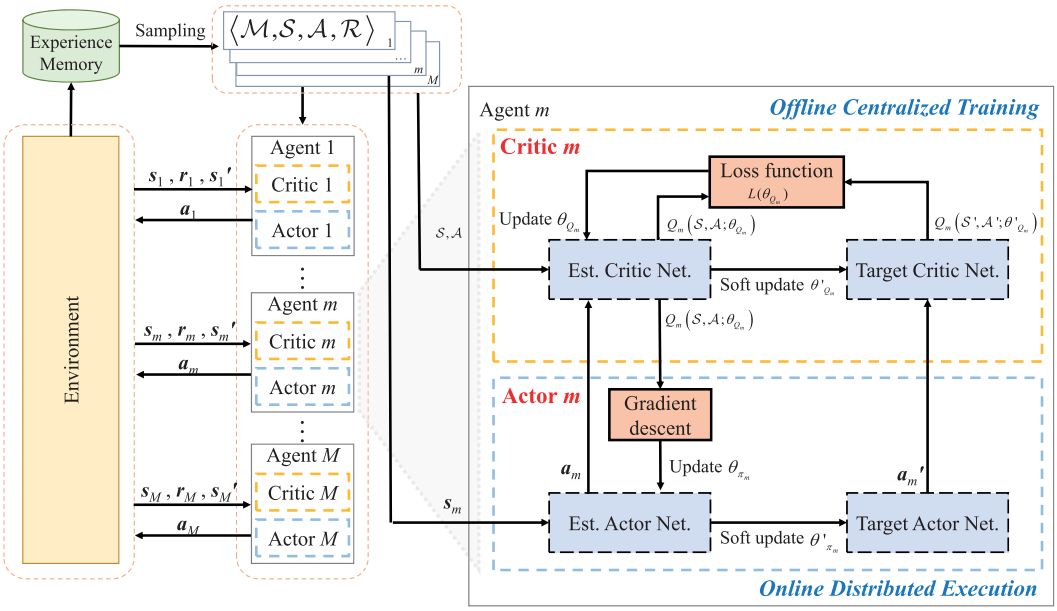



Fig. 2. Structure of MADRL-HTRCS.

On the above basis, we further define the long-term accumulative reward of $m$-agent as

$$R_m(t) = \sum_{t_0=0}^{t} \gamma_m^{t_0} r_m(t_0), \tag{24}$$

where $t_0$ denotes the previous time, and $\gamma_m \in [0,1]$ denotes the discounted factor indicating how the past reward impacts the current reward for $m$-agent.

By maximizing the long-term accumulative reward of each agent, DT can obtain an optimal task-resource collaborative scheduling policy that minimizes JCT.

$$\max\ R_m(t) = \max \sum_{t_0=0}^{t} \gamma_m^{t_0} r_m(t_0), \tag{25}$$

## V. MADRL-Based Heterogeneous Task and Resource Collaborative Scheduling Algorithm

Generally, the reformulated MDP problem can be solved by dynamic programming when the state transition probability is known. However, it is quite difficult to obtain the state transition probability since the environment is dynamic and the agent cannot predict the next state before taking action. Moreover, there exists the state space explosion problem due to the complex coupling of optimization values of multiple agents. Thus, we employ the model-free MADRL and propose the MADRL-HTRCS algorithm to learn an optimal solution.

### A. Algorithm Design

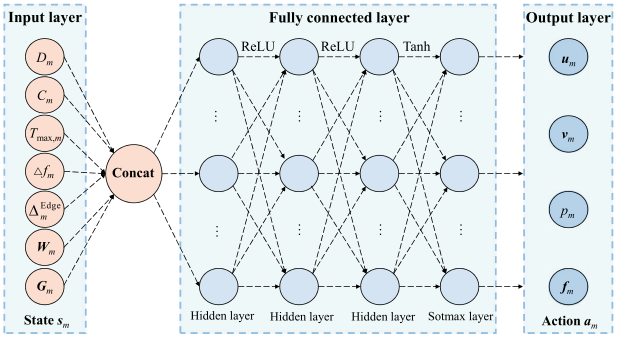
The structure of MADRL-HTRCS algorithm is depicted in Fig. 2. We employ the Actor-Critic structure as basis, where the actor is used to generate action for the agent while the critic is used to guide the actor for generating a better action. The actor further includes estimation actor network which is used for training, and target actor network which is used for action execution of agent. Similarly, the critic also includes estimation critic network and target critic network which are used to evaluate the action of actor. Herein, the actor network employs policy-based deep neural network (DNN) while the critic network employs value-based DNN.

Fully considering the dynamics of environment, we adopt the centralized training and distributed execution strategy. That is, the estimation critic network and target critic network are trained by DT in a centralized way while estimation actor network and the target actor network are executed by EDs in a distributed way.

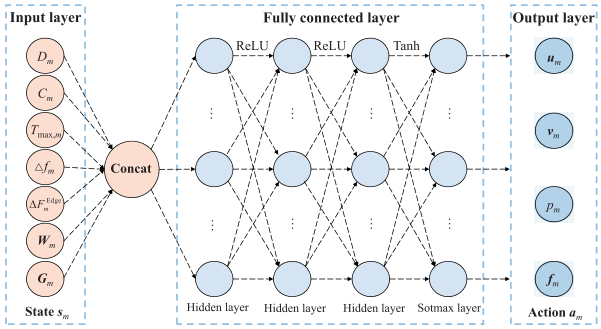
*1) Actor Network:* As shown in Fig. 3a, the actor network consists of an input layer, a fully connected layer and an output layer, where the fully connected layer includes three hidden layers and a softmax layer. For the first two hidden layers, we use the rectified linear unit (ReLU) as the activation function for nonlinear approximation. For the final hidden layer, we use tangent (Tanh) as the activation function to bound actions. In this way, the input state is transformed into all possible actions with respect to computing resource type matching decision, task division ratio, transmit power, and computing resource allocation.

For the estimation actor network of $m$-agent, the input is its current state $\boldsymbol{s}_m(t)$, indicating data size, computing resource requirement, computing resource type, estimation deviation, task deadline, bandwidth and channel power gain. After the processing of three hidden layers, the outputs are the probabilities of different actions. With the softmax layer, the sum of the output probability of each action is 1. Then, an action is selected as the final output action $\boldsymbol{a}_m(t)$.

Similarly, for the target actor network of $m$-agent, the input is the next state $\boldsymbol{s}_m(t+1)$, while the output is the next action $\boldsymbol{a}_m(t+1)$ after the processing of fully connected layer. Note that although the estimation actor network and the target actor network employ the same DNN structure, their parameters are different, which are denoted as $\theta_{\pi_m}$ and $\theta'_{\pi_m}$, respectively.




(a) Actor network




(b) Critic network

Fig. 3. Structure of actor network and critic network.

*2) Critic Network:* As shown in Fig. 3b, the critic network consists of an input layer, a fully connected layer, and an output layer, where the first two hidden layers of the fully connected layer are also associated with ReLU.

For the estimation critic network of $m$-agent, the inputs are the states and actions of all agents, namely $\mathcal{S}$ and $\mathcal{A}$. After the processing of fully connected layer, the output is the Q-value. At the decision epoch $t$, the Q-value of $m$-agent is defined as

$$Q_m\left(\mathcal{S},\mathcal{A};\theta_{Q_m}\right)=\mathbb{E}_\pi\left[R_m(t);\boldsymbol{s}(t),\boldsymbol{a}(t);\theta_{Q_m}\right]. \tag{26}$$

Similarly, for the target critic network of $m$-agent, the inputs are the next states and actions of all agents at the decision epoch $t+1$, denoted as $\mathcal{S}'$ and $\mathcal{A}'$. Correspondingly, the output is the next Q-value $Q'_m\left(\mathcal{S}',\mathcal{A}';\theta'_{Q_m}\right)$ after the processing of fully connected layer. The structures of estimation critic network and target critic network are also the same, but with parameters $\theta_{Q_m}$ and $\theta'_{Q_m}$, respectively.

### B. Algorithm Training

The centralized training of the MADRL-HTRCS algorithm is implemented by DT, as summarized in Algorithm 1. Specifically, the critic network of each agent is managed by DT which can obtain the states and actions of all agents and make them fully observable to each agent. In this way, from the perspective of one agent, the environment is static no matter what action is taken by other agents.

During the centralized training, DT first gets a global view on the states and actions of all agents, and then utilizes the information to train the estimation critic network for each agent, with the objective of maximizing the Q-value.

**Algorithm 1** MADRL-HTRCS Algorithm Training

**Input:** $M$, $N$, $D_m$, $C_m$, $T_{\max,m}$, $P_{\max}$, $I_p$, $\sigma_n^2$, $o_m$, $o_n$, $\Delta f_m$, $\Delta f_{m,n}$, $W_{m,n}$ and $g_{m,n}$ for $m=1,\ldots,M$ and $n=1,\ldots,N$;
**Output:** $\theta'_{\pi_m}$ for $m=1,\ldots,M$;
Compute $u_{m,n}$ by (6) for $m=1,\ldots,M$ and $n=1,\ldots,N$;
Initialize discount factor $\gamma$, parameter updating rate $\eta$;
Initialize estimation and target AC networks with the parameters $\theta_{\pi_m}$, $\theta'_{\pi_m}$, $\theta_{Q_m}$, $\theta'_{Q_m}$ for $m=1,\ldots,M$;
**repeat**
  **repeat**
    Initialize $\boldsymbol{s}_m(t)$;
    Input $\boldsymbol{s}_m(t)$ to estimation actor network, and obtain $\boldsymbol{a}_m(t)=\pi_m\left(\boldsymbol{s}_m(t);\theta_{\pi_m}\right)$;
    Execute $\boldsymbol{a}_m(t)$ on $\boldsymbol{s}_m(t)$, compute reward $r_m(t)$ and obtain $\boldsymbol{s}_m(t+1)$;
    Store $\left(\boldsymbol{s}_m(t),\boldsymbol{a}_m(t),r_m(t),\boldsymbol{s}_m(t+1)\right)$ as an experience in the memory for replaying;
    Input $\mathcal{S}$ and $\mathcal{A}$ to estimation critic network, and compute $Q_m\left(\mathcal{S},\mathcal{A};\theta_{Q_m}\right)$;
    Input $\mathcal{S}'$ and $\mathcal{A}'$ to target critic network, and compute $Q'_m\left(\mathcal{S}',\mathcal{A}';\theta'_{Q_m}\right)$;
    Calculate Q-value by (27);
    Calculate temporal difference error $\delta$ and the loss function $L(\theta_{Q_m})$;
    Update parameter $\theta_{Q_m}$ by stochastic gradient descent as (28);
    Input $\boldsymbol{s}_m(t)$ to estimation actor network, and obtain $\boldsymbol{a}_m(t)=\pi_m\left(\boldsymbol{s}_m(t);\theta_{\pi_m}\right)$;
    Input $\boldsymbol{s}_m(t+1)$ to target actor network, and obtain $\boldsymbol{a}_m(t+1)=\pi'_m\left(\boldsymbol{s}_m(t+1);\theta'_{\pi_m}\right)$;
    Update parameter $\theta_{\pi_m}$ by gradient descent as (29);
    Update $\theta'_{\pi_m}$ and $\theta'_{Q_m}$ by (30) and (31), respectively;
  **until** *Agent* $m=M$;
**until** *Episode* $k=K$;

For $m$-agent, its Q-value $Q_m\left(\mathcal{S},\mathcal{A};\theta_{Q_m}\right)$ is updated according to the Bellman criterion [33] as

$$Q_m\left(\mathcal{S},\mathcal{A};\theta_{Q_m}\right)=R_m(t)+\gamma_m\max Q'_m\left(\mathcal{S}',\mathcal{A}';\theta'_{Q_m}\right). \tag{27}$$

In this way, the temporal difference error is calculated as $\delta=Q'_m\left(\mathcal{S}',\mathcal{A}';\theta'_{Q_m}\right)-Q_m\left(\mathcal{S},\mathcal{A};\theta_{Q_m}\right)$ and the loss function is given by $L(\theta_{Q_m})=\mathbb{E}\left(\delta^2\right)$. Then, we update the parameter $\theta_{Q_m}$ by minimizing the loss function, wherein the stochastic gradient descent algorithm is adopted as follows.

$$\nabla_{\theta_{Q_m}}L(\theta_{Q_m})=\mathbb{E}\left[2\delta\nabla_{\theta_{Q_m}}Q_m(\mathcal{S},\mathcal{A};\theta_{Q_m})\right]. \tag{28}$$

In contrast, the actor network of each agent is deployed in ED since the actor is locally executable and can take actions according to its locally observed state. Note that it is also necessary to train the local observed states by DT which can periodically synchronize the trained DNN parameters to

**Algorithm 2** MADRL-HTRCS Algorithm Execution

**Input:** $M$, $\theta_{\pi'_1}, \ldots, \theta_{\pi'_M}$ ;
**Output:** $\boldsymbol{a}_1(t), \ldots, \boldsymbol{a}_M(t)$ ;
Initialize $\epsilon_0$, $\beta$, $m = 1$ ;
**repeat**
  Input $\theta_{\pi'_m}$ to the target actor network of $m$-agent ;
  $m$-agent observes the state $\boldsymbol{s}_m(t)$ ;
  Compute $\epsilon$ according to (33) ;
  Choose a probability $\Pr_m \in [0, 1]$ ;
  **if** $\Pr_m < \epsilon_m$ **then**
    Randomly select an action $\boldsymbol{a}_m(t) \in \mathcal{A}$ as $\boldsymbol{a}_m^{\text{Exploration}}(t)$ ;
  **else**
    Compute the value of action by $\boldsymbol{a}_m(t) = \pi'_m\left(\boldsymbol{s}_m(t); \theta_{\pi'_m}\right)$ ;
    Select the action with the maximum Q-value as $\boldsymbol{a}_m^{\text{Exploitation}}(t)$ ;
**until** $m = M$;

all agents. Herein, the parameter $\theta_{\pi_m}$ is updated by gradient descent as

$$\nabla_{\theta_{\pi_m}} L(\theta_{\pi_m}) \approx \mathbb{E}\left[\nabla_{\theta_{\pi_m}} \log \pi_m(\boldsymbol{s}_m(t); \theta_{\pi_m})\, Q_m(\mathcal{S}, \mathcal{A}; \theta_{Q_m})\right], \tag{29}$$

where $\pi_m\left(\boldsymbol{a}_m(t); \theta_{\pi_m}\right)$ indicates the policy by taking action $\boldsymbol{a}_m(t)$.

To ensure the stability of training process, we softly update the parameters of target AC network by the historical parameters of the estimation AC network as follows.

$$\theta'_{Q_m} = \eta\theta_{Q_m} + (1-\eta)\theta'_{Q_m}, \tag{30}$$
$$\theta'_{\pi_m} = \eta\theta_{\pi_m} + (1-\eta)\theta'_{\pi_m}, \tag{31}$$

where $\eta \in [0, 1]$ is the parameter updating rate.

### C. Algorithm Execution

After the centralized training by DT, EDs perform distributed execution as summarized in Algorithm 2. Specifically, $m$-agent first downloads the training results by DT and inputs them into its own target actor network. Then, $m$-agent observes the environment and state $\boldsymbol{s}_m(t)$, and generates its action $\boldsymbol{a}_m(t)$ for the reward $r_m(t)$ according to the trained policy $\pi_m$.

Initially, the agent takes actions randomly for exploration since there is not enough knowledge. When the knowledge is enough, the agent takes actions to maximize its reward. Thus, there always exists a tradeoff between exploration and exploitation, where too many explorations will affect the stability of long-term Q value calculation while too many exploitations will cause the insufficient exploration of the action space. The conventional greedy algorithm simply selects the optimal action for reward maximization, resulting in the loss of some efficient actions and the corresponding knowledge. Thus, we propose a step-by-step $\epsilon$-greedy algorithm to balance exploration and exploitation as follows.

$$\boldsymbol{a}_m(t) = \begin{cases} \boldsymbol{a}_m^{\text{Exploration}}(t), & \epsilon \\ \boldsymbol{a}_m^{\text{Exploitation}}(t), & 1-\epsilon, \end{cases} \tag{32}$$

with

$$\epsilon = (1-\beta)^K \epsilon_0, \tag{33}$$

where $\boldsymbol{a}_m^{\text{Exploration}}(t)$ is the exploration action randomly selected while $\boldsymbol{a}_m^{\text{Exploitation}}(t)$ is the exploitation action selected from the explored action space; $\epsilon_0$ is a positive value for initial exploration, $\beta$ is the decreasing rate of exploration, and $K$ is the training iterations. Obviously, with the iteration of training, $\epsilon$ decreases and the agent gradually transfers from exploration to exploitation. In this way, we can balance exploration and exploitation, and avoid the oscillation caused by setting a large $\epsilon$ for long-time.

### D. Algorithm Complexity Analysis

We further analyze the computational complexity of the proposed MADRL-HTRCS algorithm. The computational complexity mainly depends on the structure of neuron network and its number of parameters. As both actor network and critic network employ DNN, the computational complexity is calculated based on that of DNN. Given a DNN employing $L$ layers with $O_l$ neurons in $l$-th layer, the computational complexity is calculated as $\mathcal{O}(J) = \mathcal{O}\left(\sum_{l=1}^{L} O_l O_{l+1}\right)$ [20]. Thus, the computational complexities of actor network and critic network are calculated as $\mathcal{O}(J_a)$ and $\mathcal{O}(J_c)$, respectively.

At the centralized training stage, $M$ agents with $E$ experiences are trained for $K$ iterations, and the computational complexities of actors and critics are $\mathcal{O}_a(J_a K E^M)$ and $\mathcal{O}_c(J_c K E^M)$, respectively. The training process is offline completed by DT in the CS, which can provide sufficient computing resources.

At the distributed execution stage, each agent executes action according to the actor network, and the computational complexity of actor is calculated as $\mathcal{O}_a(J_a)$. The execution process is online completed by ED independently, and can guarantee the timeliness.

## VI. Performance Evaluation

To evaluate the performance of the proposed MADRL-HTRCS algorithm, we implement numerical experiments, and analyze the effectiveness and superiority by comparing with some benchmark algorithms in this section.

### A. Experiment Environment and Setting

*1) Learning and Training Environment:* The hardware setup includes Intel i7-13700k CPU and NVIDIA RTX4090-24G GPU, while the software environment includes TensorFlow-GPU-1.14.0 and Python-3.7.

The parameters of DNN are set as follows. For Actor, the numbers of neurons for the first and second hidden layers are respectively set to 300 and 100, while that for the third hidden layer is set according to the dimensionality of possible actions. For Critic, the numbers of neurons for the three hidden layers

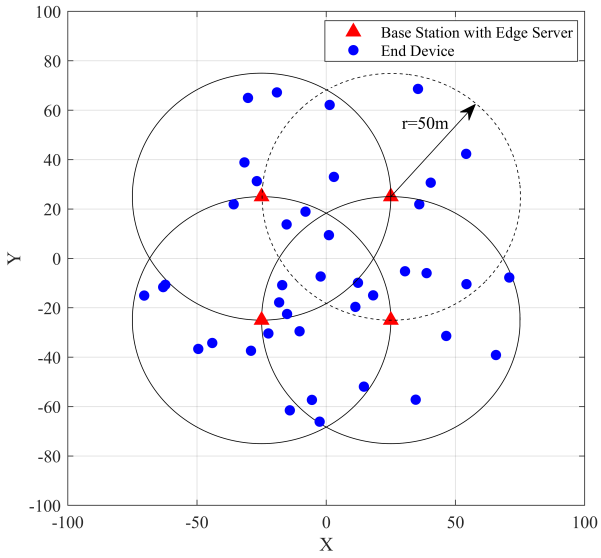



Fig. 4. Deployments of EDs and BSs with ESs.

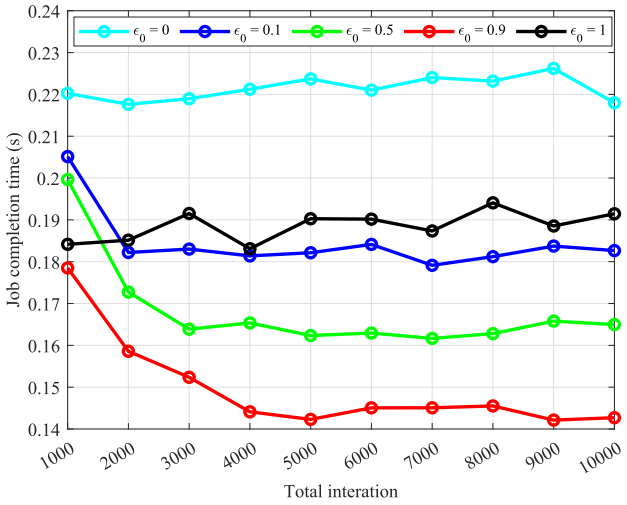



Fig. 5. JCT versus total iteration with different exploration parameters: $N = 3$, $M = 10$, $P_{\max}$=200 mw, $I_p$ =1 mW.

are set to 300, 100 and 1, respectively. During training, the learning rates of Actor and Critic networks are respectively set to $\gamma_a = 10^{-4}$ and $\gamma_c = 10^{-3}$, the discount factor is set to $\gamma_m$=0.9, the initial value and decreasing rate for exploration are set to $\epsilon_0 = 0.9$ and $\beta = 10^{-4}$, respectively [24].

*2) Network Environment and Resource Setup:* We consider a dynamic wireless network environment where the numbers of ESs and EDs are set to $N = 3 \sim 4$ and $M = 5 \sim 35$ to cover a given area. As depicted in Fig. 4, ES-enhanced BSs are fixed in given positions while EDs are randomly deployed within the coverage of BSs. By calculating the distances among EDs and ESs, we can obtain the channel power gains, where the path loss exponent is set to 3. For simplicity, the bandwidths for task offloading are equally set to $W_{m,n} =$ 20 MHz, the maximum transmit powers of EDs are equally set to $P_{\max}$=200 mW, while the noise powers are equally set to $\sigma_n^2 = 10^{-11}$ mW [20]. To evaluate the influence of interference constraints, the tolerable peak interference powers of EDs are equally set to $I_p = 10^{-6} \sim 10^6$ mW.

Without loss of generality, we assume there are only CPU and GPU computing resources randomly configured to ESs during experiments. The maximum computing resources of ESs are equally set to $F_{\max,n}$=100 GHz/s, while those of EDs are equally set to $F_{\max,m} = 5$ GHz/s. The computing resource estimation deviations of DT for ESs and EDs are randomly set to $[-0.5, 0.5]$ GHz/s.

*3) Heterogeneous Task Setup:* We consider three kinds of tasks, namely control task, sensing task, and multimedia task. The control tasks have small data size with $D_m \in [10, 300)$ Bytes and strict deadline $T_{\max,m} = 10$ ms, the sensing tasks have medium data size with $D_m \in [300, 1000)$ Bytes and medium deadline $T_{\max,m} = 50$ ms, while the multimedia tasks have big data size with $D_m \in [1000, 1500)$ Bytes and slack deadline $T_{\max,m} = 100$ ms. Correspondingly, we set $\rho_m = 300, 200, 100$ for control, sensing and multimedia tasks, respectively. During experiments, control, sensing and multimedia tasks are randomly generated and their ratios to the total tasks are around 25%, 25% and 50%, respectively. The required computational cycles for different types of tasks are equally set to $C_m$=0.25 MHz/Byte.

*4) Benchmark Algorithms for Comparison:* Fully considering existing works on partial or binary task offloading to single or multiple ESs by single-agent DRL and MADRL algorithms, we consider the following benchmark algorithms. For fair comparison, the DNN structures and parameters are set the same.

- MADRL-HTRCS: The proposed MADRL-HTRCS algorithm supporting total, partial and none task offloading to single and/or multiple ESs.
- MADRL-PSES: A MADRL-based algorithm supporting **P**artial task offloading to a **S**ingle **ES**.
- MADRL-BSES: A MADRL-based algorithm supporting **B**inary task offloading to a **S**ingle **ES**.
- DDQN-HTRCS: A single-agent DDQN-based algorithm supporting total, partial and none task offloading to single and/or multiple ESs.

## *B. Performance Evaluation*

In Fig. 5, we first evaluate the convergence of MADRL-HTRCS by the step-by-step $\epsilon$-greedy algorithm. Obviously, JCT by different exploration parameters converges around different values. When the exploration parameter is set to 0 or 1, JCT oscillates in a certain interval. This is because the $\epsilon$-greedy algorithm keeps on randomly choosing actions or exploring state space, respectively. In contrast, when there are both exploration and exploitation (i.e., $\epsilon_0 \neq \{0, 1\}$ ), JCT first decreases and then converges. For the same iteration, the larger exploration value, the smaller JCT. This is because a good action space can be established by sufficient exploration. To be specific, when the exploration parameters are 0.1 and 0.9, the total iterations required for convergence to approximate the minimum JCT are around 2000 and 4000, respectively. Thus, without loss of generality, we set $\epsilon_0 =$ 0.9 in the following experiments as there is a good balance between exploration and exploitation.

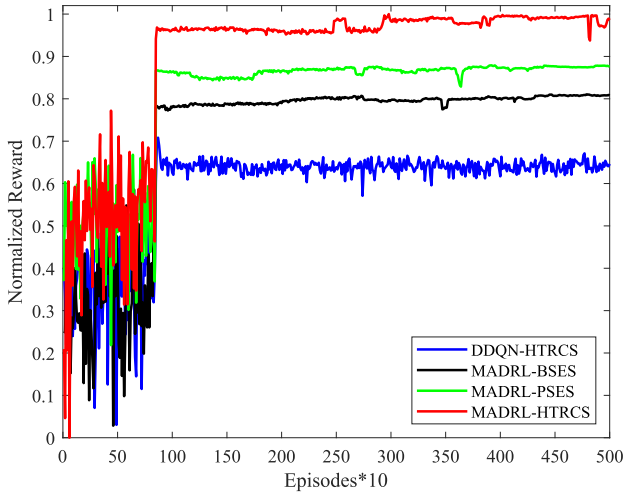



Fig. 6. Normalized reward versus iteration for different scheduling algorithms: $N = 3$, $M = 20$, $P_{\max}$=200 mw, $I_p$ =1 mW.

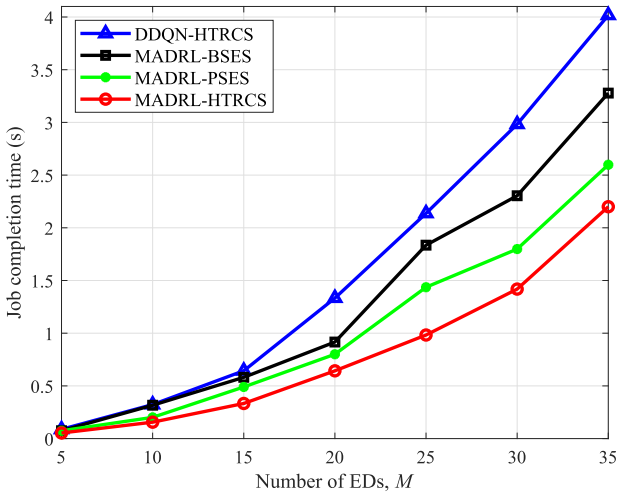



Fig. 7. JCT versus the number of EDs for different scheduling algorithms: $N = 3$, $P_{\max}$=200 mw, $I_p$ =1 mW.

Furthermore, Fig. 6 compares the normalized reward of MADRL-HTRCS with those of MADRL-PSES, MADRL-BSES and DDQN-HTRCS. With the increase of training iterations, the normalized rewards of all scheduling algorithms increase from small values to large values, and maintain at stable intervals, respectively. That is to say, all scheduling algorithms can converge, which validates their effectiveness by DRL. When these scheduling algorithms converge, the three MADRL-based algorithms obtain higher normalized rewards than the single-agent DDQN-HTRCS algorithm. This phenomenon validates the superiority of MADRL in distributed execution, while its complexity for centralized learning is not higher than that of single-agent DRL. In particular, MADRL-HTRCS obtains the highest normalized reward. This is because MADRL-HTRCS can offload partial task to multiple ESs for parallel computing according to the computing resource utilization of each ES and the channel states among EDs and ESs. In contrast, MADRL-PSES can only offload partial task to a single ES, while MADRL-BSES can only offload the whole task to a single ES or process the task locally. In this way, the utilization of computing and communication resources by MADRL-PSES and MADRL-BSES are not as sufficient as those by MADRL-HTRCS.

Fig. 7 compares JCT by different scheduling algorithms for different numbers of EDs. When the number of EDs is small (e.g., $M$=5), indicating that the job is not complex, JCT by any number of ESs and/or by any scheduling algorithms is almost the same with a small value. This is because the computing and communication resources are sufficient for EDs' tasks. With the number of EDs increasing, JCT of all algorithms increases. The reason is explained as follows. When more EDs participate in the job, namely the job becomes more complex, EDs must compete for the given computing and communication resources, resulting in the increase of both computing latency and communication latency. When the number of EDs becomes large (e.g., $M$=35), the performance gaps among different scheduling algorithms become large. In particular, JCT of DDQN-HTRCS is the largest while that of MADRL-HTRCS is the smallest. This phenomenon

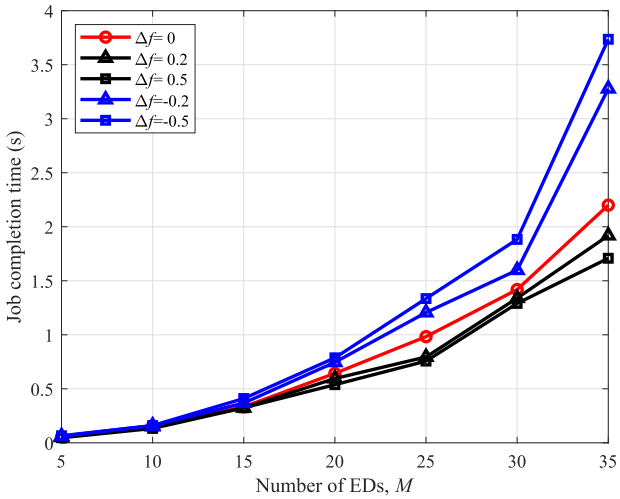



Fig. 8. JCT versus the number of EDs for different estimation deviations: $N = 3$, $P_{\max}$=200 mw, $I_p$ =1 mW.

validates that MADRL-HTRCS is more suitable for massive heterogeneous tasks collaborations.

More specifically, Fig. 8 evaluates how the estimation deviation of DT influences JCT by MADRL-HTRCS, where $\Delta f \triangleq \Delta f_n = \Delta f_{m,n} = \{-0.5, -0.2, 0, 0.2, 0.5\}$ are selected to make the figure clear. With the increase of estimation deviation, JCT decreases correspondingly. Specifically, when the estimation deviation is positive, JCT is smaller than that without estimation deviation (i.e., $\Delta f = 0$). This is mainly because the required computing resources are over estimated, and more computing resources are allocated for actual computing, which certainly reduces the computing latency and the corresponding JCT. On the contrary, when the estimation deviation is negative, the computing resources actually allocated are less than the required computing resources, which increases JCT.

Fig. 9 further presents how the number of ESs influences JCT. Obviously, for given number of EDs, JCT decreases with the increase of ESs. This is because the computing resources are enhanced with more ESs deployed, and the computing

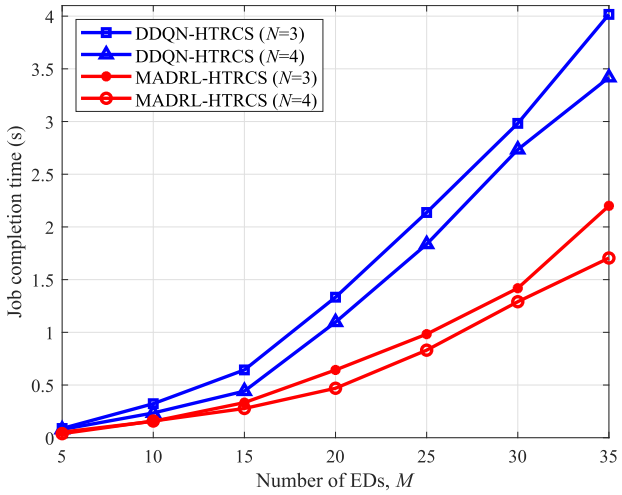



Fig. 9. JCT versus the number of EDs for different numbers of ESs: $P_{\max}$=200 mw, $I_p$ =1 mW.

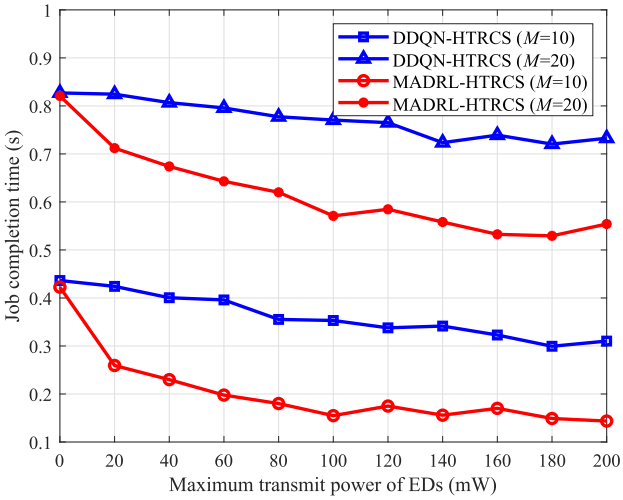



Fig. 10. JCT versus the maximum transmit power of EDs for different numbers of EDs: $N = 3$, $I_p$ =1 mW.

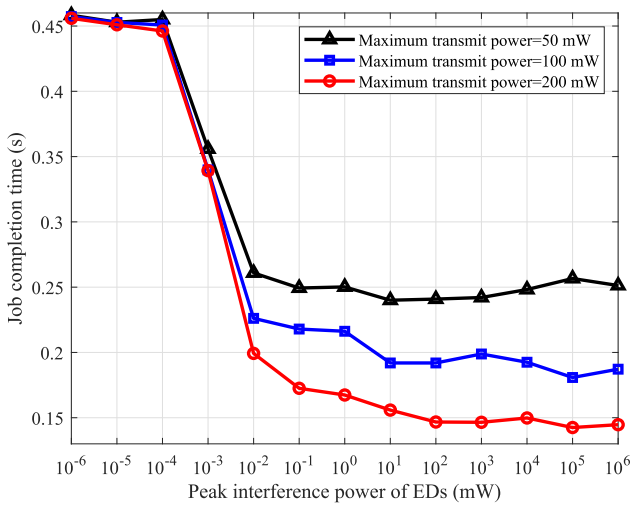



Fig. 11. JCT versus the tolerable peak interference power of EDs for different maximum transmit powers of EDs: $N = 3$, $M = 10$.

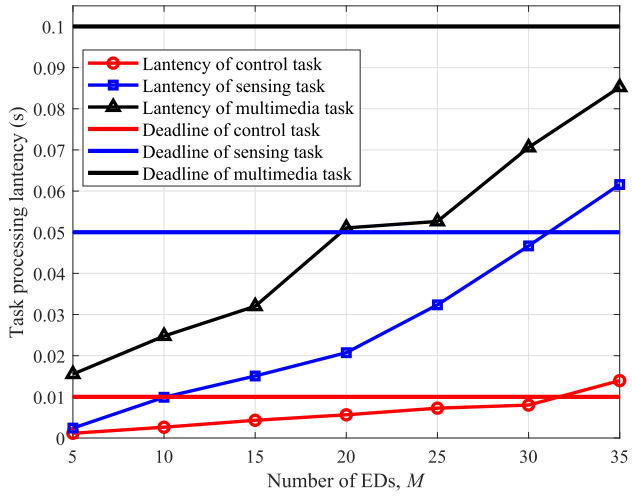


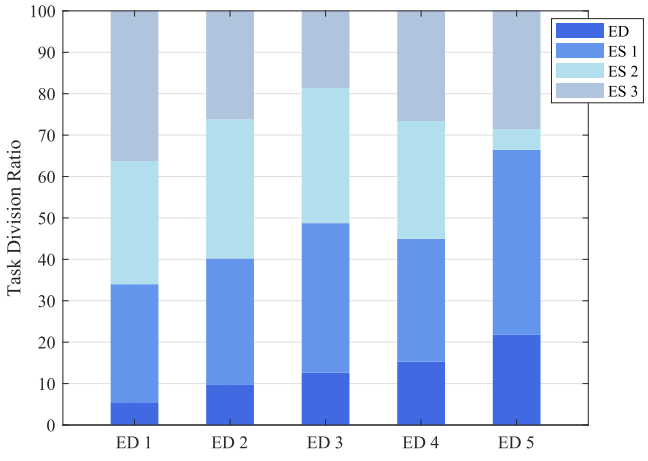

Fig. 12. Task processing latency versus the number of EDs under given deadline constraints: $N = 3$, $P_{\max}$=200 mw, $I_p$ =1 mW.

latency is reduced accordingly. Meanwhile, MADRL-HTRCS always obtains a smaller JCT than DDQN-HTRCS.

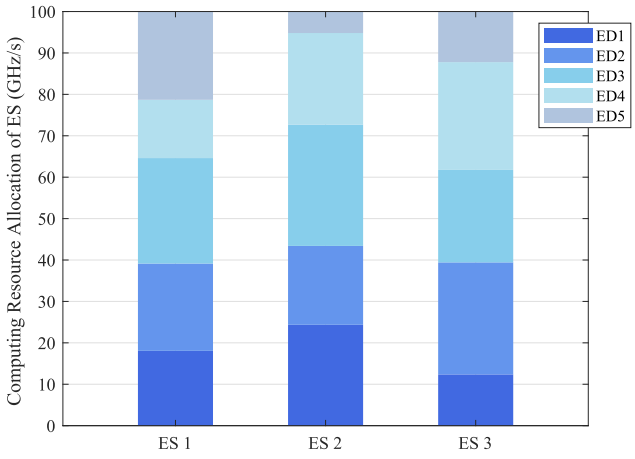
Fig. 10 demonstrates how the maximum transmit power of EDs influences JCT for different numbers of EDs. When the maximum transmit power of EDs is 0 mw, EDs do not offload any tasks to ESs for edge computing, and process tasks totally based on local computing resources. Thus, for the same number of EDs, JCT by MADRL-HTRCS and DDQN-HTRCS is the same but very large. With the maximum transmit power of EDs increasing, JCT decreases. This is because EDs can offload tasks to ESs under the peak interference power constraints. However, when the maximum transmit power of EDs achieves certain values, JCT no longer decreases. This is due to the fact that the transmit powers of EDs achieve the tolerable peak interference power, and cannot be further enhanced in order to protect other EDs. In addition, for given maximum transmit power of EDs, JCT increases with the numbers of EDs increasing due to the same reason as Fig. 7. Also, MADRL-HTRCS obtains better performance than DDQN-HTRCS.

Fig. 11 comprehensively investigates the impacts of the maximum transmit power and the peak interference power of EDs on JCT. When the peak interference power of EDs is very small (e.g., $I_p = 10^{-6}$ mw), JCT is almost equally large, regardless of the maximum transmit power of EDs. This is because the transmit powers of EDs are too small to approach 0 mw since they are strictly constrained by the peak interference power of coexisting EDs. With the increase of peak interference power of EDs, namely relaxing the interference constraint, JCT decreases since EDs can employ suitable transmit powers to offload tasks to ESs. When the peak interference power achieves certain values, JCT doesn't decrease accordingly since the transmit powers of EDs are no longer constrained by the peak interference power but constrained by the maximum transmit power. Specifically, when the peak interference power is around $I_p = 10^{-2}$ mw, JCT for $P_{\max} = 50$ mW first doesn't decrease since the transmit powers of EDs achieve their maximum values. In contrast, JCT for $P_{\max} = 100$ mW and $P_{\max} = 200$ mW can further decrease but successively converge later.

In detail, Fig. 12 further presents the average processing latencies of heterogeneous tasks by MADRL-HTRCS. We can observe that the task processing latency of each type also increases with the number of EDs, and can well satisfy the




(a) Task division radio of each ED for multiple ESs




(b) Computing resource allocation of each ES for multiple EDs

Fig. 13. Task division radios of EDs and computing resource allocations of ESs: $N = 3$, $M = 5$, $P_{\max}$=200 mw, $I_p$ =1 mW.

differentiated deadline requirements even for $M = 30$. Herein, the task processing latency of control task is the smallest, while that of multimedia task is the largest. These observations validate that DT employing MADRL-HTRCS can well schedule the heterogeneous computing and communication resources according to the requirements of heterogeneous tasks. When the number of EDs further increases (e.g., $M = 35$), the latencies of sensing and control tasks no longer satisfy their required deadlines. This is mainly because their resources requirements cannot be well satisfied when massive EDs compete for the given computing resources of ESs. For this case, if we want to guarantee the requirements of heterogeneous tasks, more resources should be deployed, such as deploying more ESs with more computing resources.

Correspondingly, Fig. 13 depicts the status of tasks division ratios of EDs and computing resource allocations of ESs by MADRL-HTRCS. We can observe that the computing resources allocation of each ES in Fig. 13(b) is generally proportional to the task division radio of each ED in Fig. 13(a). This is due to the task-oriented and on-demand resource scheduling by MADRL-HTRCS, which can divide task, control transmit power, match computing resource type and allocate computing resources according to the deadline requirements, offloading interferences and channel states.

## VII. Conclusion

In this paper, we proposed a DT-driven edge-end collaborative scheduling algorithm for heterogeneous tasks and resources based on MADRL. With full consideration of deadline requirements of heterogeneous tasks, heterogeneous computing resource types and capabilities of EDs and ESs, computing resource estimation deviation of DT, maximum transmit power and tolerable peak interference power of EDs, we formulated the JCTM problem to divide tasks for parallel computing, match the type of edge-end computing resource, allocate computing resources of ESs, and control transmit powers of EDs. Due to the non-convexity of the JCTM problem, we transferred it into a multi-agent MDP problem, where a compound reward consisting of latency reward and deadline reward was designed. Then, we employed MADRL to deal with the explosive state space and proposed the MADRL-HTRCS algorithm to approximate the optimal solution. With extensive experiments, we minimized JCT through offline centralized training by DT and online distributed execution by EDs. The results showed that, MADRL-HTRCS can satisfy the deadlines of heterogeneous tasks and achieve the smallest JCT comparing with typical benchmark algorithms.

## References

[1] X. Kong, Y. Wu, H. Wang, and F. Xia, "Edge computing for Internet of everything: A survey," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23472–23485, Dec. 2022.

[2] C. Xu, H. Yu, P. Zeng, and Y. Li, "Towards critical industrial wireless control: Prototype implementation and experimental evaluation on URLLC," *IEEE Commun. Mag.*, early access, Feb. 7, 2023, doi: 10.1109/MCOM.009.2200648.

[3] X. Liu, C. Xu, H. Yu, and P. Zeng, "Deep reinforcement learning-based multichannel access for industrial wireless networks with dynamic multiuser priority," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7048–7058, Oct. 2022.

[4] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," White paper, 2014, pp. 1–7. [Online]. Available: https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf

[5] 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.

[6] 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.

[7] H. Ahmadi, A. Nag, Z. Khar, K. Sayrafian, and S. Rahardja, "Networked twins and twins of networks: An overview on the relationship between digital twins and 6G," *IEEE Commun. Standards Mag.*, vol. 5, no. 4, pp. 154–160, Dec. 2021.

[8] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13789–13804, Sep. 2021.

[9] F. Tang, X. Chen, T. K. Rodrigues, M. Zhao, and N. Kato, "Survey on digital twin edge networks (DITEN) toward 6G," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1360–1381, 2022.

[10] S. Bi, L. Huang, H. Wang, and Y. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.

[11] I. Khan, X. Tao, G. M. S. Rahman, W. U. Rehman, and T. Salam, "Advanced energy-efficient computation offloading using deep reinforcement learning in MTC edge computing," *IEEE Access*, vol. 8, pp. 82867–82875, 2020.

[12] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.

[13] Z. Ning et al., "Deep reinforcement learning for intelligent Internet of Vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cognit. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.

[14] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4968–4977, Jul. 2021.

[15] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5G services in mobile edge computing systems: Learn from a digital twin," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4692–4707, Oct. 2019.

[16] T. Zhang, Z. Wang, Y. Liu, W. Xu, and A. Nallanathan, "Caching placement and resource allocation for cache-enabling UAV NOMA networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12897–12911, Nov. 2020.

[17] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mumtaz, "Efficient offloading for minimizing task computation delay of NOMA-based multiaccess edge computing," *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3186–3203, May 2022.

[18] V. D. Tuong, W. Noh, and S. Cho, "Delay minimization for NOMA-enabled mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 7321–7331, Oct. 2022.

[19] C. Zhou et al., "Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 911–925, Feb. 2021.

[20] 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.

[21] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5098–5107, Jul. 2021.

[22] V. D. Tuong, T. P. Truong, T.-V. Nguyen, W. Noh, and S. Cho, "Partial computation offloading in NOMA-assisted mobile-edge computing systems using deep reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13196–13208, Sep. 2021.

[23] K. Wang, Y. Zhou, Z. Liu, Z. Shao, X. Luo, and Y. Yang, "Online task scheduling and resource allocation for intelligent NOMA-based industrial Internet of Things," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 803–815, May 2020.

[24] X. Liu, C. Xu, H. Yu, and P. Zeng, "Multi-agent deep reinforcement learning for end—Edge orchestrated resource allocation in industrial wireless networks," *Frontiers Inf. Technol. Electron. Eng.*, vol. 23, no. 1, pp. 47–60, Jan. 2022.

[25] S. Chen, J. Chen, Y. Miao, Q. Wang, and C. Zhao, "Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 8, pp. 364–375, 2022.

[26] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng, "Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1427–1444, Jan. 2022.

[27] K. Zhang, J. Cao, and Y. Zhang, "Adaptive digital twin and multi-agent deep reinforcement learning for vehicular edge computing and networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1405–1413, Feb. 2022.

[28] J. Cai, H. Fu, and Y. Liu, "Multitask multiobjective deep reinforcement learning-based computation offloading method for industrial Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1848–1859, Jan. 2023.

[29] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, Sep. 2021.

[30] H. Lu, X. He, M. Du, X. Ruan, Y. Sun, and K. Wang, "Edge QoE: Computation offloading with deep reinforcement learning for Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9255–9265, Oct. 2020.

[31] X. Xu et al., "Service offloading with deep Q-network for digital twinning-empowered Internet of Vehicles in edge computing," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1414–1423, Feb. 2022.

[32] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*. Berlin, Germany: Springer, 1978.

[33] N. C. Luong et al., "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

**Chi Xu** (Senior Member, IEEE) received the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2017. He is currently a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China. His research interests include industrial wireless networks, 5G URLLC, edge computing, and tactile internet. He is a Voting Member of IEEE 1918.1 working group for tactile internet and a member of IEEE 1932.1 working group for licensed/unlicensed spectrum interoperability in wireless mobile networks. He also serves as a standardization delegate for 3GPP TSG RAN.

**Zixuan Tang** received the B.E. degree from the Shengyang University of Technology, Shenyang, China, in 2021. She is currently pursuing the master's degree with the University of Chinese Academy of Sciences, Beijing, China. Her research interests include industrial wireless networks and artificial intelligence.

**Haibin Yu** (Senior Member, IEEE) received the Ph.D. degree from Northeastern University, Shenyang, China, in 1997. He has been a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, since 1997. He has published three books, authored or coauthored more than 200 papers, and held more than 50 patents. He and his research team have proposed the WIA-PA and WIA-FA standards which are specified as IEC 62601 and IEC 62948, respectively. His research interests include industrial communication and networked control, industrial automation, and intelligent manufacturing. He was elected as a fellow of ISA for his contributions in Fieldbus technologies in 2011. He serves as the Chair for China National Technical Committee for Industrial Process Measurement Control and Automation Standardization and the Vice-Chair of Chinese Association of Automation.

**Peng Zeng** received the Ph.D. degree from the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, in 2005. He has been a Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences, since 2007. He is currently an Expert Member of the IEC/TC65/WG16, a member of the Standards Committee of SP100, and a member of the Wireless WG of FieldBus Foundation. He also serves as the Chair for Edge Computing Technical Committee, Chinese Association of Automation. His research interests include wireless sensor networks and industrial wireless communications.

**Linghe Kong** (Senior Member, IEEE) received the B.S. degree from Xidian University in 2005, the M.S. degree from Telecom SudParis in 2007, and the Ph.D. degree from Shanghai Jiao Tong University in 2013. He is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He was a Post-Doctoral Researcher with Columbia University, McGill University, and the Singapore University of Technology and Design. His research interests include the Internet of Things, wireless networks, big data, and mobile computing.