Joint Task and Computing Resource Allocation in Distributed Edge Computing Systems via Multi-Agent Deep Reinforcement Learning

Yan Chen, Yanjing Sun, Hao Yu, and Tarik Taleb, *Senior Member, IEEE*

***Abstract*—Edge servers can collaborate to enhance service capability. However, cloud servers may be unable to execute centralized management due to unpredictable communications. In such systems, distributed task and resource management are vital but challenging due to heterogeneity and various restrictions. Therefore, this paper studies such edge systems and formulates the distributed joint task and computing resource allocation problem for maximizing the quality of experience (QoE). Given the restrictions on real-time state observations and resource management involving other facilities, we decompose it into subproblems of distributed task allocation and computing resource allocation. After formulating the problem as a partially observed Markov decision process, we propose a two-step approach that depends on multi-agent (MA) deep reinforcement learning. First, each edge server performs a policy to allocate tasks for its associated users according to a partial observation. We employ the MA deep deterministic policy gradient to tackle vast spaces of discrete actions. Besides, we incorporate the action entropy of massive users' task allocation to enhance exploration. Then, we prove that the QoE-maximized computing resource allocation is a problem of maxing a sum of sigmoids, and we address it by sigmoidal programming. Simulation results reveal that the proposed approach dramatically improves the system QoE and reduces the average service latency. Besides, the proposed solution outperforms benchmarks in training and convergence.**

***Index Terms*—Edge computing, distributed task allocation, resource allocation, quality of experience, multi-agent deep reinforcement learning**

I. Introduction

EDGE computing (EC) can improve quality of service (QoS) for ever-changing applications by extending the ability of clouds to network edges [1]–[3]. Collaboration among edge servers (ESs) is critical for fully utilizing the limited resources of edge networks and enhancing their serviceability [4]–[7], but it requires efficient orchestration of resources and tasks among ESs. However, interactions between ESs and the remote cloud server (RCS) suffer from unpredictable communications, making the RCS unable to provide reliable centralized management of tasks and resources in the edge networks. Furthermore, ESs are unable to provide central management due to their limited resources and restricted access to the prior knowledge, real-time state, and resource allocation of other ESs, primarily due to information-exchanging costs like privacy risks and handshaking time. As a result, users or ESs in such systems must conduct task allocation and resource management in a distributed manner under partial state observations.

In a distributed EC system, different components can infer and implement task allocation or resource allocation decisions independently. For example, the facilities initiating computing tasks (e.g., users and service providers) [8]–[14] or the infrastructures supporting the EC service (i.e., ESs) [15]–[19]. However, facilities in a distributed system may be unable to access the global state due to restrictions like association and authority, forcing them to make decisions based on partial state observations. In addition, under a random system state, the state transition and performance only depend on the real-time system state, actions jointly taken by all facilities, and the system's transition features. Therefore, the problem is generally formulated as a partially observed Markov decision process (POMDP) in existing works devoted to distributed task allocation or resource allocation. Moreover, traditional methods generally require information exchange to fetch information about other components and obtain a result through considerable iterations. As a result, they are hardly implemented in a highly dynamic distributed EC system. Recently, multi-agent deep reinforcement learning (MADRL) is increasingly being exploited to address POMDP problems as it can tackle challenges like system heterogeneity, time-varying states, and restricted state observation. MADRL and its variations have presented great potential in systems where multiple facilities are working cooperatively or competitively with limited sharing of information [20], [21]. However, most existing works on distributed task allocation assume that users and ESs are endowed with access to some global state, which

This work was supported in part by the National Key Research and Development Program of China under Grant No. 2021YFB2900200; in part by the National Natural Science Foundation of China under Grant No. 62071472; This work was also conducted at ICTFICIAL Oy, Finland; and supported in part by the European Union's Horizon 2020 Research and Innovation Program through the Charity project under Grant No. 101016509; in part by the AerOS project funded by the European Union's Horizon Europe, the EU's key funding program for research and innovation under Grant No. 101069732; in part by the Fundamental Research Funds for the Central Universities under Grant No. 2020ZDPY0304; and in part by the Future Network Scientific Research Fund Project of Jiangsu under Grant No. FNSRFP-2021-YB-34.*(Corresponding author: Yanjing Sun)*

Yan Chen is with Zhejiang Lab, Hangzhou, 311121 China. He was with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, 221116 China. (e-mail: yanchen_edu@outlook.com)

Yanjing Sun is with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, 221116 China. (e-mail: yjsun@cumt.edu.cn)

Hao Yu is with ICTFICIAL Oy and was with the Center of Wireless Communications, University of Oulu, Oulu, 90570 Finland. (e-mail:hao.yu@oulu.fi)

Tarik Taleb is with Ruhr University Bochum, Bochum, 44801 Germany. (e-mail: tarik.taleb@rub.de)

Manuscript received **, 2023; revised ** , 2024.

is unattainable in some actual systems. In addition, training a policy for every user in an EC system with massive users is challenging and may exhaust resources.

Moreover, compared with the quality of service (QoS) like latency, the quality of experience (QoE) is more relative to users' actual feelings and better reflects the service level of modern user-centered applications [22]. Although QoE is positively correlated with QoS, they are generally non-linearly correlated [23], [24]. The most significant feature is the existence of saturated areas where the user's QoE will not significantly change with the variation of QoS. For example, the QoE of a user is extremely satisfied when experiencing high QoS, like ultra-low latency. Although allocating more resources further reduces the latency, the user feels an insensible QoE increase. In such conditions, allocating more resources to improve such users' QoE is meaningless compared to allocating these resources to improve the QoE of other users. Consequently, allocating resources according to the QoE imposes greater significance in improving resource utilization and satisfying more users, especially in EC systems with resource restrictions.

Existing works have contributed to distributed task allocation in various EC systems. However, most of them consider users to conduct task allocation with the assumption that users can obtain the resource state of all ESs, and ESs accept the resource requirements proposed by users [9]–[12], [25]. In actual systems, accessing the resource states and allocations of ESs may be infeasible for users. Besides, considering the limited state observation of users and ESs, most existing works have proposed MADRL-based approaches that enable users or ESs to conduct distributed management. However, training a policy for every user is non-trivial in a system with massive users, which hinders similar approaches to working in mass-user EC systems. Therefore, this paper proposes that ESs conduct task and resource allocation in a distributed manner. This satisfies the restriction in authority to access resource state and allocation and significantly reduces the number of policies, thereby reducing the complexities and costs associated with policy training. Although a few works also enabled ES to conduct management, they assume that agents can obtain the states of all users [15], [16] and ES would accept the resource allocation decisions suggested by others without considering the resource competition between users [17], [19]. This paper studies scenarios where each ES can only observe the state of its real-time associated users, and each ES allocates its resources to served users independently rather than accepting propositions from other agents. Besides, there is resource competition between users (i.e., resources concurrently consumed by different users cannot overlap). To address these restrictions, we propose a two-step MADRL-based approach that includes task allocation and resource allocation to maximize the overall QoE for users. Here, we summarize the main contributions as follows:

- We study an EC system with massive users, in which users and ESs cannot access the real-time state and resource allocation of other ESs. Then, we formulate the distributed joint task and computing resource allocation problem (DJTCRA) for maximizing the sum of users' QoE by considering the quantitative correlation between QoE and QoS.
- Next, we propose utilizing MADRL to tackle challenges regarding system heterogeneity, dynamic task requests, and limited state observations. Specifically, considering users and ES cannot access the real-time state and resource allocation of other ESs, as well as the difficulties in training policies for massive users, we set ES as the controller. Then, according to state availability, we model DJTCRA as a POMDP. Meanwhile, we decompose the problem into a distributed task allocation problem and a QoE-maximizing resource allocation problem.
- Subsequently, we developed a two-step solution to satisfy these restrictions. An approach that leverages the policy-based MADRL is employed to realize massive user task allocation under partial state observation. Meanwhile, we exploit sigmoidal programming to optimize the computing resource allocation following each task allocation after proving it is a problem of maximizing a sum of sigmoids. Moreover, we further integrate the entropy of massive users' task allocation actions into training multi-agent policies to enhance exploration.
- Simulation results reveal that our proposed approach can efficiently establish collaboration among ESs in a distributed EC system supporting massive users. Meanwhile, it outperforms benchmarks in system QoE and service latency by jointly optimizing task and resource allocation.

The rest of this paper is organized as follows: Related works are reviewed in Section II. Section III introduces the system model. Section IV formulates the investigated DJTCRA problem. Section V details the proposed MADRL-based DJTCRA approaches. Simulations are conducted and discussed in Section VI. This paper is concluded in Section VII.

II. Related works

Researchers have widely investigated task and resource management in centralized EC systems, promoting the development of the EC continuum [26], [27]. Unlike centralized systems, distributed EC systems lack a central controller to gather global information and manage the system. Meanwhile, facilities in distributed systems have difficulties obtaining other devices' information due to reasons like security or unacceptable costs of exchanging information (e.g., latency). Therefore, increasing research efforts have been devoted to distributed task allocation or resource allocation to mine and utilize the advantages of collaborations among ESs in various distributed EC systems.

Edge task allocations are typically formulated as variations of combinatorial optimization problems. Traditional mathematical approaches can hardly solve these issues alone, especially when the system is heterogeneous and highly dynamic. Therefore, further considering the restrictions inherent in distributed EC systems, approaches based on traditional decentralized algorithms have been developed. Moreover, as AI technologies have presented great potential, multi-agent reinforcement learning has been widely explored to solve such complex problems in distributed EC systems and has shown advantages over traditional approaches.

TABLE I
COMPARISON BETWEEN RELATED WORKS AND GAP ANALYSIS

Ref.	Agent	Method	Parallel execution	Flexible allocation	ES collaboration	Global state				Obj	Resource competition	Access authority	Limitations		
						1	2	3	4				Specfic	Common	
[28]	User	Matching theory	✓	✗	✓		✓			utility	✓	✘			
[29]		Game theory	✗	NA	✓		✓	✓	✓	latency	NA	✘		$\star^2$	
[30]	ES	Game theory	✗	NA	✗		✓			utility	NA	✘			
[8]	Users	MAQL	✓	✓	✓			✓		latency	✓	✘	$\star^4$		
[9]		MA A2C	✗	NA	✓		✓			cost	NA	✘			
[10]		MADDPG	✓	✗	✓	✓	✓			utility	✓	✘		$\star^3$	$\star^1$
[11]		MADDPG	✓	✗	✓	✓	✓			cost	✘	✘			
[12]		MADDPG	✓	✗	✗	✓				cost	✘	✘			
[15], [16]	ES	MADDPG	✗	NA	✓			✓		latency	NA	✘	$\star^5$		
[17]		MADQN	NA	NA	✓			✓	✓	reward	✘	✘			
[31]		MA AC	✓	✓	✓		✓			reward	✓	✘	$\star^6$		
[18], [19]		MADDPG	✓	✓	✓					PE	✘	✘			
This work		MADDPG + Action Entropy	✓	✓	✓					QoE	✓	✓			

$\star^1$- Assume allow interactions or the existence of a central entity to gather and share the states of other components before making each decision.
$\star^2$- It requires considerable iterations before obtaining a decision for any given state, which is time-consuming.
$\star^3$- When implemented in systems with massive users, the training of policies is challenging and may exhaust resources.
$\star^4$- It can only handle finite state and action spaces obtained through discretization; available action needs to be filtered manually in advance.
$\star^5$- All ESs are homogeneous and configured with several task queues, and each task must allocated to one queue;
- It cannot decide the allocations of each task but only migrates loads of task queues to other ESs under the coordination of a central entity.
$\star^6$- Each time, multiple ESs may grab the same task to execute, resulting in collisions and a waste of resources.

Global state: **1**←BS, **2**←ES, **3**←task, **4**←user; **NA**-not applicable

A. Approaches based on Traditional Methods

In [28], a distributed task allocation solution based on matching theory is studied. They develop a heuristic algorithm that allocates tasks iteratively according to tasks' preference lists. Each preference list comprehensively considers the ES' computing power, the conditions of the wireless channel, and the time limits. In [29], a game theoretical model of distributed task allocation was formulated. Then, a variational inequality theory was utilized to obtain the task allocation equilibrium strategy in static mixed strategies. After that, they developed a decentralized algorithm that supports users in making their offloading decisions with the assistance of information sent from a central entity periodically. A mean-field game model was created for the decentralized load schedule among ESs in [30]. Then, they developed a pricing scheme based on Lyapunov optimization to distribute computing load to ensure long-term utility and load balancing.

However, these traditional decentralized approaches generally require information exchange or obtaining information from a virtual central data collector before each decision, which introduces costs for exchanging information and may be infeasible in actual systems. Besides, most of these approaches solve the optimization problems under each given state via iterations to approach an optimal solution, which is usually time-consuming. Thus, it is hard to meet the real-time management requirements of highly dynamic EC systems.

B. Approaches based on MA Reinforcement Learning

In [8], an MA Q-learning (MAQL) was employed, which enables each service provider initiating the task requests to allocate its tasks to one of the multiple associated ESs. However, such an approach can only address problems with limited state and action space, which hinders its applications in complex systems. In [9], every user can obtain all ESs' real-time computing load states and decide its task allocation via a policy based on the advantage actor-critic. Multi-agent deep deterministic policy gradient (MADDPG) has been a popular MADRL method in recent years. It enables each policy to adapt to other policies by utilizing a centralized training method with combined states. As a result, MADDPG has been exploited in some distributed task allocation projects [10]–[12], [25]. These works assume each user is an agent to perform a MADDPG-based policy. Then, they can make decisions independently for their task and resource allocations with the input of local state observation and the state of all available resources (e.g., resource states of all ESs [11], [12] or communication states of all BS [10], [25]). Nevertheless, these approaches empower users to perform policies, but they require the assistance of some global state information, which imposes certain constraints on their practical implementation. First, requiring global information is infeasible in some systems since users may lack the authorization to access the resource states of all ESs or BSs. Meanwhile, devices or controllers with privileges (e.g., BS and ES) are typically responsible for resource allocation rather than users. In addition, policy training is generally resource-hungry. Consequently, training policies for every user dramatically increase costs, including policy training and parameter synchronization from the RCS to users, especially in systems with massive users. Moreover, although MADDPG utilizes combined states for policy training to reduce the impact of agents' high dynamics, it remains challenging when training large numbers of policies.

Instead of users, a few recent works select ESs to execute distributed management. In [15], [16], the MADDPG-based policy enables each ES to actively select some tasks to process from a given task set based on its resource state. However, they assume there are a limited number of tasks in the system, and

every ES can observe the information about all tasks waiting to be served. In [17], MA deep Q-learning was exploited to optimize the resource rental decisions of an application service provider (ASP). The instantiation of the ASP on each ES determines the amount of resources to rent according to the demand received from the served users. They assume the task offloading has been given and the available resources on each server are discrete. Therefore, this approach can only be utilized for problems with a small action space and cannot determine each user's task and resource allocations. In [31], a DDPG-extended MA actor-critic (AC) training method is developed. Each ES executes a policy to allocate computing resources to a limited number of task queues they maintain and migrate part of their computing loads to other ESs In [18], [19], tasks and resources are allocated to optimize the designed processing efficiency (PE) in Cybertwin IoT by MADDPG. However, the resource competition between tasks processed by the same ES has not been well studied.

A comparison of the most related works discussed above is detailed in Table I. We first focus on comparisons: Does the work allow ES to process different tasks in parallel? Can computing resources be allocated flexibly according to requirements? Can ES establish collaboration to share loads? Then, we list if the proposed approach requires some global state about other components. In addition, we discuss whether they studied the resource competition between tasks and the access authority of agents. We treat access authority as whether a device is authorized to make decisions relevant to others, such as resource allocation. Different from previous works, this paper considers mass-user EC systems where ES only acquires the state of real-time associated users. Consequently, each ES must independently allocate tasks and resources for only their associated users under limited state observation. This is more reasonable since frequent information exchange is time-consuming and privacy-risky. Besides, compared with user-executed approaches, it can significantly reduce the number of policies that need to be trained since ESs are generally much fewer than users. Different from [19], we consider the resource competition between tasks. Meanwhile, each ES cannot decide the resource allocation for others because infeasible actions would break the resource constraints. Besides, they may have customized resource allocation schemes. In addition, we look into improving the QoE after considering its quantitative relationship with the QoS. We solve the resource allocation problem using sigmoidal programming. Furthermore, we integrate the action entropy of massive users' task allocation into policy training for the investigated multi-agent problem. This is beneficial for dealing with huge discrete action spaces and improving exploration.

III. System Model

A. Preliminary

As shown in Fig. 1, this paper considers a system consisting of multiple radio access networks (RAN) established by base states (BSs). Each RAN is directly associated with an ES. Thus, we use $\mathcal{H}$ to represent the set of RANs and ESs. Meanwhile, massive mobile users ($\mathcal{U}$) are performing computation-intensive applications, and each user moves randomly in the system and connects to one RAN at any time through wireless links. ESs can communicate with each other via links among RANs to establish a cooperation space for task allocation. On top of the edge network, RCS can interact with ESs to collect system states but suffers from high latency.

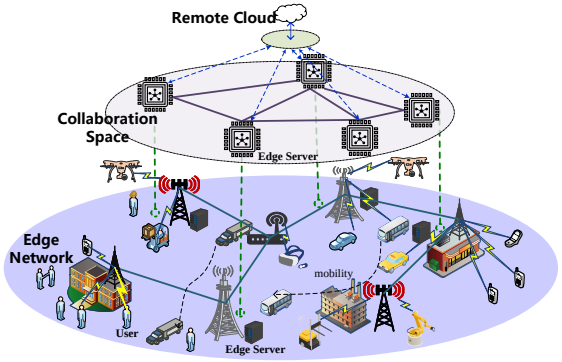


Fig. 1. A distributed edge computing system

Users always offload their tasks to the ES associated with its connected RAN. After receiving a request, the ES allocates the task to one ES in the system. A user's task is further offloaded to the RCS if it can achieve higher QoS. After completing the task processing, a result is fed back to the user via the reverse path. Although resource-limited ESs collaborate to support these applications, each ES can only decide the task and resource allocation for its associated users (i.e., the users connecting to the RAN or whose tasks are allocated to the ES) for reasons like privacy risk. In addition, RCS is responsible for more complex duties like training the policies executed on each ES based on the experiences gathered from ESs.

B. Communication Model

This paper assumes users communicate over orthogonal sub-channels and do not interfere with each other. Besides, the RAN can allocate multiple sub-channels to one user. The communication latency from a user u to the connected BS depends on the channel condition, i.e.,

$$d_u^c(t) = \frac{V_u^T(t) + V_u^R(t)}{\mathcal{B}_u(t)\log_2\left(1 + \mathcal{I}_u(t)g_u(t)/\mathcal{N}_u(t)\right)}, \tag{1}$$

where $V_u^T(t)$, $V_u^R(t)$, and $\mathcal{I}_u(t)$ are respectively u's task volume (i.e., data size), result volume, and transmitting power at time t. $g_u(t)$ and $\mathcal{N}_u(t)$ are respectively the channel gain (path loss) and noise power between u and its associated RAN. $\mathcal{B}_u(t)$ is the bandwidth allocated to u. We employ the path loss model for 5G in rural scenarios [32], i.e.,

$$PL = \begin{cases} PL_1, & 10\text{m} \le \Upsilon_{2d} \le \Upsilon_{BP}, \\ PL_2, & \Upsilon_{BP} \le \Upsilon_{2d} \le 10\text{km}, \end{cases} \tag{2}$$

where

$$\begin{aligned} PL_1 =& 20\lg(40\pi\Upsilon_{3d}f_c/3) + \min(0.03\tilde{h}^{1.72}, 10)\lg\left(\Upsilon_{3d}\right) \\ & - \min(0.044\tilde{h}^{1.72}, 14.77) + 0.002\lg(\tilde{h})\Upsilon_{3d}, \end{aligned} \tag{3}$$

and

$$PL_2 = PL_1\left(\Upsilon_{BP}\right) + 40\lg(\Upsilon_{3d}/\Upsilon_{BP}). \tag{4}$$

Υ_{2d} is the 2-dimensional (2D) ground distance from a user to the connected BS. Υ_{3d} is the 3D distance between their antennas. $\tilde{h}$ is the average building height. Υ_{BP} is the break-point distance, i.e.,

$$\Upsilon_{BP} = 2\pi \mathbb{H}_{BS} \mathbb{H}_{UT} f_c / c, \tag{5}$$

where $\mathbb{H}_{BS}$ and $\mathbb{H}_{UT}$ separately denote the height of BSs and users. f_c is the central frequency, and its unit is separately GHz and Hz in (3) and (5). $c = 3 \times 10^8$ m/s.

The task processing of a user may be allocated to a remote ES, which introduces the forwarding latency from the connected RAN i to the corresponding ES j, i.e.,

$$d_u^f(t) = \frac{V_u^T(t) + V_u^R(t)}{\mathcal{B}_{i,j}} + 2\delta_{i,j}, \tag{6}$$

where $\mathcal{B}_{i,j}$ represents the efficient communication rate of the link established between RAN i and RAN j, and $\delta_{i,j}$ is the propagation latency between them.

C. Computation Model

This paper employs a popular computation model in the EC community [2], i.e., the computing latency depends on the requirement and the allocated resource, i.e.,

$$d_u^p(t) = \frac{V_u^T(t)\kappa_u(t)}{p_u(t)}, \tag{7}$$

where $\kappa_u(t)$ is the computing intensity of u at time t, which indicates the CPU cycles required to process a per-bit task. $p_u(t)$ represents the cycles allocated to process u's task.

In addition, ESs have different resource and performance features when processing different applications. For example, an ES is optimized for processing video applications by equipping it with GPUs. Therefore, we introduce an additional factor to reveal the performance difference when the tasks of an application are processed on different ESs, i.e.,

$$d_u^p(t) = \sum_{h \in \mathcal{H}} y_u^h(t) \eta_u^h \frac{V_u^T(t)\kappa_u(t)}{p_u(t)}, \tag{8}$$

where $y_u^h(t) = 1$ indicates u's task processing is allocated to ES h at time t, $y_u^h(t) = 0$ otherwise. When u's task is allocated to ES h, the multiplicative factor η_u^h is used to measure the difference that exists in the task processing latency compared with the benchmark. Without loss of generality, the η_u^h of a user on each ES is independent and only determined by the ES ability and the tasks of the user.

IV. Problem Formulation

The EC service latency includes the wireless communication latency, the forwarding latency, and the task processing latency. Therefore, the service latency of u under a given task and resource allocation policy at time t is

$$D_u(t) = d_u^c(t) + d_u^f(t) + d_u^p(t). \tag{9}$$

Compared with QoS, QoE is more accurate in reflecting a user's personalized satisfaction. Better QoE indeed benefits from better QoS, but QoE and QoS are generally quantitatively correlated and present an exponential relationship. Researchers usually model such a relationship using a sigmoid function Fig. 2(a) [22], [24], [33], as shown in. This paper also employs such a function to model the QoE of each user u, i.e.,

$$\Omega_u(t) = \frac{L_u}{1 + e^{-\alpha_u(t)(\xi_u(t) - \beta_u(t))}}, \tag{10}$$

where $\Omega_u(t)$ and $\xi_u(t)$ are respectively u's QoE and QoS. L_u is the maximum QoE that u can enjoy. $\beta_u(t)$ is the middle point correlation coefficient, which reflects the sensitive area where the QoE changes fast as the QoS changes. $\alpha_u(t)$ is another positive coefficient (i.e., $\alpha_u(t) > 0$) that reflects the change speed correlation between QoE and QoS. Fig. 2 and (10) reveal that the QoE of a user only changes quickly around the middle point as the QoS changes. The region where the user's QoE is not satisfied but the QoS has a significant impact is what we refer to as the user-disturbed area. The user's QoE is sufficiently satisfied when the QoS is high enough and does not noticeably improve as the QoS rises. Similarly, the QoE is severely disturbed when the QoS falls below a lower threshold, forcing the user to give up the provided service, and the QoS deterioration causes a negligible decline in QoE. The QoE decreases with QoS reduction and approaches zero.

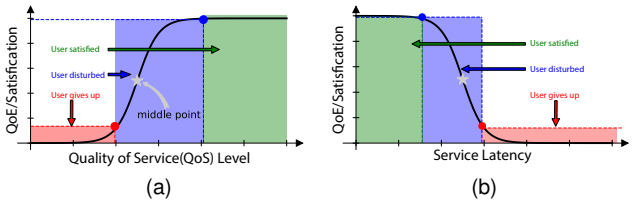


Fig. 2. The correlation between QoE and QoS

This paper selects service latency as the QoS, which is generally negatively correlated with QoE, i.e., a lower latency leads to a better QoE. Thus, we omit the negative symbol in (10) and express the QoE of a user u as [23]

$$\Omega_u(t) = \frac{L_u}{1 + e^{\alpha_u(t)(D_u(t) - \beta_u(t))}}, \tag{11}$$

where $\beta_u(t)$ is a reference latency value, around which u experiences significant QoE variation when $D_u(t)$ changes. Fig. 2(b) is a graphical example of the correlation between QoE and service latency expressed in (11).

From the above analysis, we conclude that when a user's QoE reaches the QoE-satisfied area, investing additional resources cannot improve its QoE significantly. Instead of investing more resources to improve the QoS of QoE-satisfied users, we can allocate more resources to QoE-unsatisfied users and increase the sum of users' QoE (system QoE). Therefore, allocating resources based on QoE has more practical significance in resource-limited EC systems. Therefore, this paper aims to maintain the long-term performance of maximizing the sum of users' QoE in a distributed EC system by jointly optimizing the task and computing resource allocation, i.e.,

$\mathbf{y}, \mathbf{p} = \{\mathbf{y}_u, p_u | u \in \mathcal{U}\}$, for any observed state. Then, the QoE-optimized DJTCRA problem in this paper is formulated as

$$\mathbf{P1:} \max_{\mathbf{y},\mathbf{p}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \sum_{u\in\mathcal{U}} \Omega_u(t), \tag{12}$$

$$s.t. \quad \mathbf{C1:} \sum_{h\in\mathcal{H}} y_u^h(t) \leq 1, \forall u \in \mathcal{U}, \forall t, \tag{13}$$

$$\mathbf{C2:} \sum_{u\in\mathcal{U}} y_u^h(t) p_u(t) \leq \mathcal{P}^h, \forall h \in \mathcal{H}, \forall t, \tag{14}$$

$$\mathbf{C3:} \; p_u(t) \geq 0, \forall u \in \mathcal{U}, \forall t, \tag{15}$$

$$\mathbf{C4:} \; y_u^h(t) \in \{0,1\}, \forall u \in \mathcal{U}, \forall h \in \mathcal{H}, \forall t, \tag{16}$$

where **C1** and **C4** indicate that each user's tasks can only be allocated to one ES at most. **C2** indicates that the resource allocated to users by an ES h cannot exceed the maximum capacity ($\mathcal{P}^h$). **C3** reveals the resource allocated to a user must be non-negative. All symbols defined till now are listed in TABLE II. To simplify the presentation, we ignore the explanation of the symbol (t), i.e., at time t.

TABLE II
LIST OF DEFINED NOTATIONS

Symbol	Definition
$\mathcal{U}$	The set of users in the EC system
$\mathcal{H}$	The set of RAN nodes and ESs
$\mathcal{B}_{i,j}$	The communication rate between RAN nodes i and j
$\delta_{i,j}$	The propagation delay between RAN nodes i and j
$\mathcal{P}^h$	The maximum computing resource capacity of ES h
$\mathcal{B}_u(t)$	Wireless communication bandwidth allocated to user u
$\mathcal{I}_u(t)$	Transmission power of user u
$\mathcal{N}_u(t)$	Wireless communication noise of user u
$g_u(t)$	Wireless communication channel gain of user u
Υ_{2d}	2D distance between a user and the connected RAN
Υ_{3d}	3D communication distance between a user and RAN
Υ_{BP}	Break-point distance in the communication path loss model
f_c	The central frequency of communication spectrum
$\mathbb{H}_{BS}$	The height of base stations in the system
$\mathbb{H}_{UT}$	The height of users' devices
$\tilde{h}$	The average height of buildings in the cover of the system
$V_u^T(t)$	The volume of computing tasks generated by user u
$V_u^R(t)$	The volume (data size) of u's computing results
$p_u(t)$	Computing resource (CPU cycles) allocated to user u
κ_u	Computing intensity (cycles/bit) of u's task processing
η_u^h	Performance difference factor of u's tasks processing
$y_u^h(t)$	Binary indicator indicating if u's task is allocated to ES h
$d_u^c(t)$	Wireless communication latency in the EC process of u
$d_u^f(t)$	Forwarding latency in the EC process of u
$d_u^p(t)$	Task processing latency in the EC process of user u
$D_u(t)$	The service latency of the whole EC process of user u
$\Omega_u(t)$	The QoE of the user u
L_u	The maximum QoE level can be achieved by user u
$\xi_u(t)$	The real-time actual QoS value of u
$\beta_u(t)$	The middle point coefficient of u's QoE-QoS correlation
$\alpha_u(t)$	The coefficient reflecting the QoE variation speed of u

V. MADRL-BASED DJTCRA APPROACHES

A. Problem Analysis and Decomposition

This work considers EC systems with massive users. As a result, problem **P1** is challenging, especially in the following concerns: First, each user and ES has no prior knowledge about other ESs and has limited authority to decide the task allocation and resource allocation of others. Besides, ES and users are heterogeneous regarding resource capacities, computing abilities, and task requests. Moreover, users' location, task requests, and wireless communication conditions are time-varying. Therefore, the traditional methods make it difficult to address these complexities and provide reliable management in such systems. In addition to task allocation, resource allocation is critical to optimizing QoE. However, as an ES only obtains a user's state after establishing an association, resource allocation is hardly optimized simultaneously with task allocation when considering resource allocation restrictions. The reason is that resource allocation decisions from other agents for tasks processed on an ES may not satisfy the resource capacity constraints (i.e., **C2**). This would introduce additional complexity regarding whether to accept the resource allocation decision for involved users.

At any time the system runs, there are two associations between RANs and users, i.e., communication association (CA) and computation association (PA). At any time slot, the CA users of a RAN node connect to the RAN. The PA users of an ES are users whose task processing is allocated to the ES after task allocation. In a distributed EC system, CA users of a RAN node depend on users' mobility, and only their connected RAN can access their real-time state. However, an ES can only identify the PA users after implementing a task allocation. Besides, after task allocation, the performance only depends on the resource allocation of every ES. Then, the overall QoE can be maximized by optimizing the sum of users' QoE on every ES via optimized resource allocation at this stage. Thus, based on such an order, we decompose the DJTCRA problem into two steps: 1) distributed task allocation for CA users completed by each BS, and 2) resource allocation on each ES for PA users following the task allocation.

B. QoE-Maximizing Computing Resource Allocation

At any time slot, users' communication and forwarding latency are determined after implementing a task allocation. Meanwhile, each ES can identify its PA users and acquire corresponding information, including real-time computing requirements and QoE-related features. Then, each user's QoE only depends on the computing resource allocation because the communication latency, computing requirements, and QoE properties have been fixed. Moreover, the maximum system QoE can be achieved by maximizing the sum QoE of all PA users of every ES because each ES works independently. Therefore, after task allocation, the optimal QoE is obtained by optimizing computing resource allocation on every ES. We use $\mathcal{U}_h(t)$ to represent the set of an ES h's PA users after implementing task allocation at time slot t. According to (1), (6), (8), (9), and (11), the QoE-maximized resource allocation problem on the ES h can be written as can be expressed as

$$\mathbf{P2:} \max_{\boldsymbol{p_u}} \sum_{u\in\mathcal{U}_h} \frac{L_u}{1+e^{\alpha_u(d_u^c+d_u^f+\eta_u^h \frac{V_u^T \kappa_u}{p_u}-\beta_u)}}, s.t. \; \mathbf{C2}, \mathbf{C3}, \tag{17}$$

where we ignore the time symbol t. Besides, **C1** and **C4** are naturally disappeared after task allocation. Only the resource capacity and the resource allocation value constraints need to be satisfied when an ES allocates resources to its PA users. **C2**

and **C3** are both linear constraints. The feasible set of p_u is an obvious convex set and is a *polyhedron* [34]. Besides, the QoE function of each user in **P2** is differentiable and monotonically increasing with respect to the variable p_u. Moreover, we have

Proposition 1: There exists a value $\epsilon > 0$, which makes the QoE function of a certain user up concave when $p_u \geq \epsilon$.

Proof 1: The proof is attached in the Appendix I.

Thus, **P2** is an NP-hard problem of maximizing a sum of sigmoids [35]. Then, we can resort to sigmoidal programming to obtain the QoE-maximized computing resource allocation. We use the bisection method to roughly select the inflection points of QoE functions [36].

C. Framework of MADRL-based DJTCRA Approach

The method above can allocate the computing resource following a task allocation. Then, **P1** is converted to optimize the task allocation without considering resource allocation. This is a difficult combinatorial optimization problem, especially considering system heterogeneity and limited prior knowledge. Besides, approaches requiring global knowledge are ineffective in our work because each ES independently makes decisions with limited state observation. Thus, we exploit MADRL to realize distributed task allocation. MADRL has been exploited to deal with the heterogeneity and dynamics of EC systems. However, most existing works consider a small group of users to conduct task allocation with the assumption of knowing the resource states of all ESs (e.g., [10]–[12], [25]). In addition to the difficulty of obtaining sufficient states, training a policy for each user in a system with massive users is non-trivial. In multi-agent systems, each agent's policy is dynamically updated over the training process, introducing additional and highly dynamic complexity. At the same time, each policy needs to adapt to the dynamics of all other policies during the training process. This is arduous in a system where massive agents work in a distributed manner of cooperation or competition. Therefore, this work proposes to conduct distributed task allocation by ES. In addition to satisfying the restrictions on accessing state and resource management for other components, this approach can significantly reduce the number of policies. As a result, this approach simplifies the complexity of policy training and reduces corresponding resource requirements (e.g., synchronizing the models from the cloud to each agent and the memory required by training policies). Besides, to cope with the vast and discrete action space of massive users' task allocation, we developed an approach that relies on the policy-based MADDPG method. This involves training with policy output but executing the action mapped from policy output since naive MADDPG is designed to handle continuous action [37]. To enhance the exploration, we integrate the action entropy of massive users' actions into policy training for the multi-agent task allocation problem of massive users. In such a problem, each agent faces a vast and discrete action space.

According to the system model, we set all ESs to perform distributed task allocation, and they jointly determine the system task allocation. Assuming users' states maintain stability when implementing a DJTCRA. Then, the DJTCRA process is a POMDP since ESs work cooperatively, and each ES h can only observe its local state $\mathcal{O}_t^h$. At any time t, the system transition depends on the joint action $\mathcal{A}_t$ taken by all ESs and the current system state $\mathcal{S}_t$, i.e., $\mathcal{S}_t \times \mathcal{A}_t \rightarrow \mathcal{S}_{t+1}$. The system state is the union of all local states, i.e., $\mathcal{S}_t = (\bigcup_{h\in\mathcal{H}} \mathcal{O}_t^h) \cup \boldsymbol{s_t}, \forall t$, where $\boldsymbol{s_t}$ is the additional available state. The system action is the union of actions independently taken by every ES, i.e., $\mathcal{A}_t = \bigcup_{h\in\mathcal{H}} a_t^h, \forall t$. Based on the system setting and objective, we define states, actions, and the reward function as follows:

Local state: The partial state observation of each RAN h at the beginning of a time slot t, including the real-time properties of its CA users ($\mathcal{U}_t^h$), including the requests, QoE and QoS correlations, locations, and communication conditions, i.e.,

$$\mathcal{O}_t^h = [h, V_u^T, V_u^R, \kappa_u, \alpha_u, \beta_u, \Upsilon_{2d}^u, \mathcal{B}_u, \mathcal{N}_u]_{|\mathcal{U}_t^h|\times 9}, \quad (18)$$

in which, we ignore the time symbol t for simplification. Υ_{2d}^u is the 2D distance from u to its connected RAN node.

System state: The union of all local states, i.e.,

$$\mathcal{S}_t = [u_h, V_u^T, V_u^R, \kappa_u, \alpha_u, \beta_u, \Upsilon_{2d}^u, \mathcal{B}_u, \mathcal{N}_u]_{|\mathcal{U}|\times 9}, \quad (19)$$

where u_h indicates the RAN that u is connecting. Besides, we assume users communicate with the associated RAN on fixed transmitting power and only move on the ground surface.

In addition, we set the arrays representing the local state and the system state to share the same structure, i.e., a 2D array whose size is $|\mathcal{U}|\times 9$. Then, we can deploy policies sharing the same structure on every ES to deal with the issue of different numbers of CA users due to mobility. We assume the RCS has indexed all users according to information like MAC addresses and shares the indexes with ESs in advance. Then, each row of the array contains the state of the corresponding users. In a local state array, the rows corresponding to users who are not the CA users of the ES are filled with 0, and other rows are encapsulated with the CA users' information.

Distributed task allocation action: The action generated by an ES indicates the assignment of each CA user's tasks to one ES. For each ES, there is an enormous action space. Then, value-based methods that compare all candidates to select one are impractical when there are numerous users. Therefore, we employ a policy-based method and define each user's action as the probability of allocating its task to every ES, as in our prior work [37], so that the action maintains the policy gradient, i.e.,

$$a_t^h = [p_{i,j}]_{|\mathcal{U}_t^h|\times|\mathcal{H}|}. \quad (20)$$

$p_{i,j}$ is the probability of selecting ES j to process tasks of the i th user in $\mathcal{U}_t^h$. The i th row of a_t^h is $a_t^h[i] = [p_{i,1}, \cdots, p_{i,|\mathcal{H}|}]$ and satisfies $\sum_{j=1}^{|\mathcal{H}|} p_{i,j} = 1$. After generating a_t^h according to $\mathcal{O}_t^h$, the ES indicated by the highest probability in each row is selected to process the task of the corresponding user. The union of selected ESs is the implementable action $a_{h,t}$, i.e.,

$$a_{h,t}[i] = \arg\max\{a_t^h[i]\}, \forall i \in \mathcal{U}_t^h, \quad (21)$$

System action: The union all ESs' actions, i.e.,

$$\mathcal{A}_t = [p_{i,j}]_{|\mathcal{U}|\times|\mathcal{H}|}. \quad (22)$$

We also set distributed task allocation actions and system actions to have the same data structure. When an ES generates

an action for its CA users, it only picks data from the rows corresponding to its real-time CA users. This setting also endows the DJTCRA model with the potential to handle newly joined users by maintaining several additional state and action rows. After allocating indexes to newly joined users, extra rows can accommodate their states and corresponding actions. If detecting a policy mismatch, the latest collected experiences can be utilized for adapting policy by methods like meta-learning [38], which is beyond the scope of this work.

Reward: According to **P1**, we defined the system reward obtained after implementing an action as system QoE, i.e.,

$$\mathcal{R}_t = \sum_{u \in \mathcal{U}} \Omega_u(t) = \sum_{h \in \mathcal{H}} r_t^h, \tag{23}$$

where r_t^h is the sum QoE of the CA users of ES h.

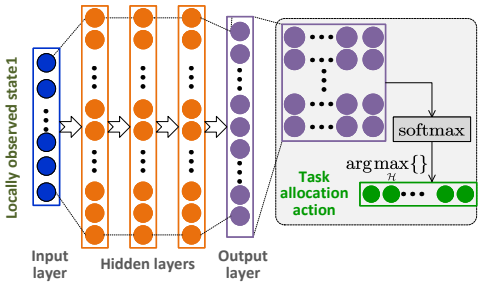


Fig. 3. The structure of a DJTCRA policy network.

We design a policy network as shown in Fig. 3, in which outputs of the last layer are reshaped to a 2D array with size $|\mathcal{U}| \times |\mathcal{H}|$. Then, each row is processed by *softmax* to produce the distributed task allocation action. Finally, we can obtain the executable action by executing the $\arg\max$ on each row.

D. Architecture for Training and Executing policies

As depicted in Fig. 4, we employ an architecture that includes centralized training on the RCS and distributed executions on ESs [39]. Considering the resource restrictions of ESs, DJTCRA policies are trained on RCS, which reduces the resource consumption of ESs. Meanwhile, the RCS maintains global information, i.e., the local state observation and distributed task allocation of all ESs. Then, utilizing system states and actions for policy training allows each policy to adapt to other ESs' policies and improve training efficiency and stability. Updated policies are synchronized from the cloud to the corresponding ESs. Then, each ES executes the updated policy to manage the system and collect more experience. Distributed execution of DJTCRA policies on ESs rather than by users dramatically reduces the costs of training and parameter synchronization. We name the controller performed on each ES as an agent, including the actor for task allocation and a resource allocation module.

The main work procedures of this DJTCRA architecture are displayed in Fig. 5. The cloud only interacts with each ES when synchronizing the actor's parameters to it and receiving collected data from it. In the beginning stage, the cloud initiates the images of all agents relevant to policy training. In addition to the image of the corresponding actor, twin critics,

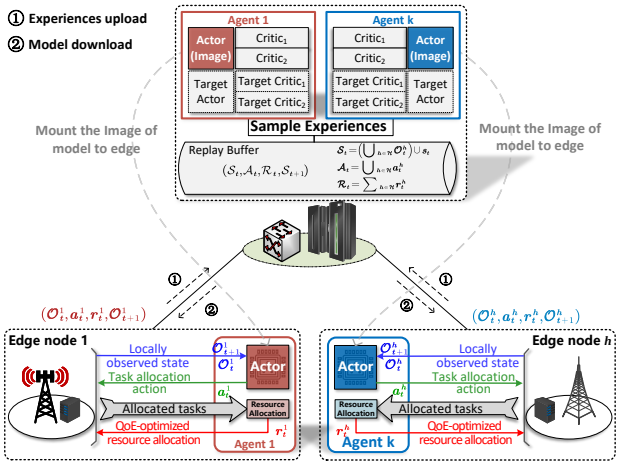


Fig. 4. The architecture for the MADRL-based DJTCRA approach

corresponding target policies, and a replay buffer for gathering data updated from ESs are created to assist in policy training. The actor image shares the same structure as that performed on ES, and they are mounted to each other for quick synchronization. After initiation and each time of updating actors, the parameters of actors are synchronized to the corresponding ES. After each time of parameter synchronization, ESs would perform the updated agent to manage the system cooperatively. The main work processes of the agent performed on each ES include task allocation and resource allocation. At each time slot, each ES observes the state of its identified CA users. Then, it will input the observed state of these users ($\mathcal{O}_t^h$) to its actor and generate corresponding task allocation action for these users (a_t^h). After the tasks of all ESs' CA users are assigned to corresponding ESs, each ES can identify its PA users and perform the resource allocation module to allocate computing resources. Finally, each ES obtains the QoE of its CA users and tracks a local state transition (r_t^h and $\mathcal{O}_{t+1}^h$). Then, each ES collects and uploads such experiences. Meanwhile, after receiving data from ES, RCS records it in the instantiated replay buffer. Once gathering sufficient data, RCS samples data from the replay buffer and updates policies. We employ twin critics and soft-delay update methods to improve training stability [40]. Then, once actors are updated, the parameters are synchronized to the corresponding agents. The above procedures are repeated until convergence. It is worth mentioning that the training process can be conducted independently after gathering enough data without always waiting for new data from ES.

E. Training of DJTCRA Policies based on MADRL

Assuming DJTCRA policies are $\boldsymbol{\pi} = \{\pi_{\theta_1}, \cdots, \pi_{\theta_{|\mathcal{H}|}}\}$ and their parameters are $\boldsymbol{\theta} = \{\theta_1, \cdots, \theta_{|\mathcal{H}|}\}$. The objective of training a DJTCRA policy is to enable it to generate distributed task allocation actions for any given local state observation at any time. The action collaborates with actions generated by other ESs to maximize the long-term expected reward, i.e.,

$$\mathcal{J}(\theta_h) = \mathbb{E}_{\{\boldsymbol{\mathcal{S}}, \pi_{\theta_h}\}}[\mathcal{R}_t | \mathcal{S}_t, (\pi_{\theta_1}(\mathcal{O}_t^1), \ldots, \pi_{\theta_{|\mathcal{H}|}}(\mathcal{O}_t^{|\mathcal{H}|}))]. \tag{24}$$

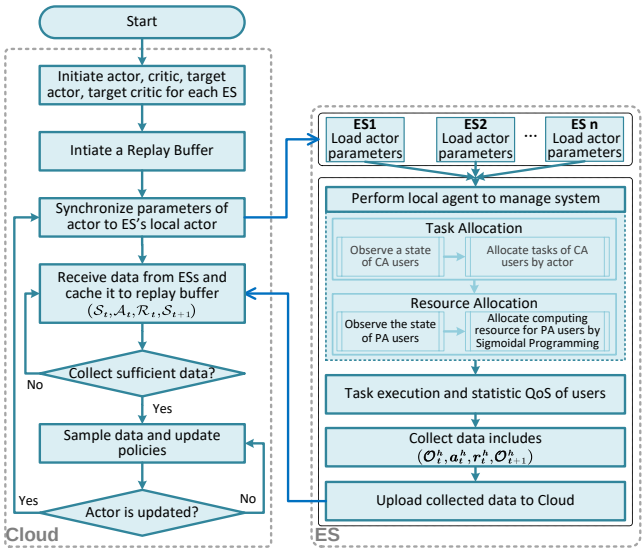


Fig. 5. The work flowchart of MADRL-based DJTCRA approach

The expected reward of a given state $\mathcal{S}_t$ after implementing an action named the state-action value or Q-value, i.e.,

$$Q_h^{\boldsymbol{\pi}}(\mathcal{S}_t,\mathcal{A}_t)=Q\left(\mathcal{S}_t,\left(\pi_{\theta_1}(\mathcal{O}_t^1),\cdots,\pi_{\theta_{|\mathcal{H}|}}(\mathcal{O}_t^{|\mathcal{H}|})\right)\right). \quad (25)$$

According to Bellman equation [41],

$$Q_h^{\boldsymbol{\pi}}(\mathcal{S}_t,\mathcal{A}_t)=\mathcal{R}_t+\gamma\mathbb{E}_{\{\mathcal{A}_{t+1}\sim\boldsymbol{\pi}\}}\left[Q_h^{\boldsymbol{\pi}}(\mathcal{S}_{t+1},\mathcal{A}_{t+1})\right], \quad (26)$$

where $\gamma\in[0,1)$ is a discount factor. Assuming the system state maintains stability when implementing a DJTCRA process, the system transition depends only on deterministic actions and system transition features. According to the policy gradient theory and MADDPG [42], the gradient of expected reward to the policy performed on ES h is

$$\nabla_{\theta_h}\mathcal{J}(\theta_h)=\mathbb{E}_{\{\boldsymbol{\mathcal{S}},a_t^h\}}\left[\nabla_{\theta_h}a_t^h\nabla_{a_t^h}Q_h^{\boldsymbol{\pi}}\left(\mathcal{S}_t,a_t^1,\cdots,a_t^{|\mathcal{H}|}\right)\right], \quad (27)$$

where $a_t^h=\pi_{\theta_h}(\mathcal{O}_t^h),\forall h\in\mathcal{H},\forall t$. We can find that the gradient calculation requires a_t^h to maintain the gradient of π_{θ_h}, which is the reason for using probabilities as the action for policy training (Section V-C). Then, the policy can be updated by minimizing the negative expectation of Q-values, i.e.,

$$\mathcal{L}(\theta_h)=\mathbb{E}_{\mathcal{D}}\left[Q\left(\mathcal{S}_t,\left(\pi_{\theta_1}(\mathcal{O}_t^1),\cdots,\pi_{\theta_{|\mathcal{H}|}}(\mathcal{O}_t^{|\mathcal{H}|})\right)\right)\right], \quad (28)$$

via the gradient descent approach. $\mathcal{D}$ is a batch of experiences $\{(\mathcal{S}_t,\mathcal{A}_t,\mathcal{R}_t,\mathcal{S}_{t+1})\}$ sampled from the replay buffer, and each $\mathcal{O}_t^h$ is extracted from $\mathcal{S}_t$. Due to system randomness, we generally estimate state-action values by a neural network in real implementations, i.e., the critic or Q-network represented by Q_{ψ_h}. The critic aimed at estimating state-action values accurately, whose parameters ψ_h can be updated by minimizing the Bellman residual loss [40], i.e.,

$$\mathcal{L}(\psi_h)=\mathbb{E}_{\mathcal{D}}\left[\left(Q_{\psi_h}(\mathcal{S}_t,\mathcal{A}_t)-\Psi(t)\right)^2\right]. \quad (29)$$

$\Psi(t)$ is the target estimation of Q-value, which is the sum of the experienced reward of executing $\mathcal{A}_t$ under state $\mathcal{S}_t$ and a discounted Q-value of $(\mathcal{S}_{t+1},\hat{\mathcal{A}}_{t+1})$, i.e.,

$$\Psi(t)=\mathcal{R}_t+\gamma Q_{\hat{\psi}_h}(\mathcal{S}_{t+1},\hat{\mathcal{A}}_{t+1}). \quad (30)$$

$\hat{\mathcal{A}}_{t+1}$ is the union of actions regenerated by the latest target actors $(\pi_{\hat{\theta}_h})$. The corresponding target critics $(Q_{\hat{\psi}_h})$ estimate the Q-values of the corresponding next state-action pairs to increase training stability.

Although the above-described MADDPG method has been widely employed in existing works [10], [12], a deterministic policy is prone to converge to sub-optimal areas [43]. Some studies have approved that introducing action entropy can significantly improve exploration abilities over the training process [44], [45]. Besides, in problems with limited discrete actions, the action entropy can be directly calculated according to the probability distribution of candidate actions [46]. Given this, we integrate max-entropy (MAxEnt) to train policies for the investigated multi-agent DJTCRA problem with vast and discrete action spaces.

To integrate action entropy, the target estimation of state-action values is converted to

$$\Psi(t)=\mathcal{R}_t+\gamma(Q_{\hat{\psi}_h}\left(\mathcal{S}_{t+1},\hat{\mathcal{A}}_{t+1}\right)+\omega_h\hat{\mathbb{S}}_{t+1}^h\Big), \quad (31)$$

where ω_h is a trainable max entropy coefficient. $\hat{\mathbb{S}}_{t+1}^h$ is the entropy of action $\hat{a}_{t+1}^h$ generated by the corresponding target actor for $\mathcal{O}_{t+1}^h$. Each user's task allocation action space includes all ESs, and users are independent. Thus, we have

$$\hat{\mathbb{S}}_t^h=-\sum_{u\in\mathcal{U}_t^h}\sum_{h'\in\mathcal{H}}\hat{p}_{u,h'}(t)\log\left(\hat{p}_{u,h'}(t)\right), \quad (32)$$

where $\hat{p}_{u,h'}(t)$ is the probability of selecting ES h' to process u's tasks in the action $\hat{a}_t^h$. Meanwhile, the loss function for updating an actor is re-defined as

$$\mathcal{L}(\theta_h)=\mathbb{E}_{\mathcal{D}}\left[Q(\mathcal{S}_t,\tilde{\mathcal{A}}_t)+\omega_h\tilde{\mathbb{S}}_t^h\right], \quad (33)$$

where $\tilde{\mathcal{A}}_t$ is the union of actions generated by the latest actors according to corresponding local states. $\tilde{\mathbb{S}}_t^h$ is the action entropy of $\tilde{a}_t^h$, which can also be calculated through (32). ω_h can be updated by minimizing the difference between action entropy and target action entropy. The target action entropy can be the negative action dimension [45]. In the investigated problem, the action dimension of each user is 1, i.e., an ES. Therefore, ω_h can be updated by minimizing

$$\mathcal{L}(\omega_h)=\mathbb{E}_{\mathcal{D}}\left[\omega_h(\sum_{u\in\mathcal{U}_t^h}(\sum_{h'\in\mathcal{H}}\tilde{p}_{u,h'}(t)\log\tilde{p}_{u,h'}(t)-1))\right], \quad (34)$$

where $\tilde{p}_{u,h'}(t)$ is obtained from $\tilde{a}_t^h$ that generated by the updated actor with $\mathcal{O}_t^h$ as the input.

Algorithm 1 illustrates the training of DJTCRA policies based on the proposed approach (MAMaxEnt). As detailed in algorithm 2, each ES works independently to determine task allocation for its CA users, resource allocation for its PA users, and upload experiences to the RCS. To provide data for policy training at the initial stage, there is a warm-up process in the system that first runs T_c episodes to collect data. After that, the training process is triggered. When updating critics, the action entropy of distributed task allocation is calculated (line 8) based on the action generated by the target actor (line 7). Then, the target estimation of the state-action value is calculated (lines 9–10). Then, the critics are updated (lines

Algorithm 1 Policy training via MAMaxEnt

Input: EC system, users, DJTCRA policies
Output: ES executable DJTCRA policies

1: Initialize a replay buffer, ω_h, and policies: actor, critic, and target policies $(\theta_h, \psi_h, \hat{\theta}_h, \hat{\psi}_h)$, $\forall h \in \mathcal{H}$
2: **for** $t' = 1, 2, 3, \cdots$ **do**
3: Collect experiences as detailed in Algorithm 2
4: **if** $t' > T_c$ **then**
5: **for** $h \in \mathcal{H}$ **do**
6: Sample $\boldsymbol{\mathcal{D}}$ and extract $\mathcal{O}_{t+1}^{h'}, \forall h' \in \mathcal{H}$
7: $\hat{\mathcal{A}}_{t+1} = \bigcup_{h' \in \mathcal{H}} \pi_{\hat{\theta}_{h'}}(\mathcal{O}_{t+1}^{h'})$
8: Calculate $\hat{\mathbb{S}}_{t+1}^{h}$ according to (32)
9: $\Psi(t) = \mathcal{R}_t + \gamma \min(Q_{\hat{\psi}_h^i}(\mathcal{S}_{t+1}, \hat{\mathcal{A}}_{t+1}))|_{i=1,2}$
10: $\Psi(t) = \Psi(t) + \gamma \omega_h \hat{\mathbb{S}}_{t+1}^{h}$
11: $\mathcal{L}(\psi_h^i) = \mathbb{E}_{\boldsymbol{\mathcal{D}}}[(Q_{\psi_h^i}(\mathcal{S}_t, \mathcal{A}_t) - \Psi(t))^2]|_{i=1,2}$
12: $\psi_h^i = \text{Adam}(\nabla_{\{\psi_h^i\}}(\mathcal{L}(\psi_h^i))|_{i=1,2}$
13: **if** $t' \% \lambda_{\mathbb{S}} == 0$ **then**
14: **for all** $h \in \mathcal{H}$ **do**
15: Sample $\boldsymbol{\mathcal{D}}$ and extract $\mathcal{O}_t^{h'}, \forall h' \in \mathcal{H}$
16: $\tilde{\mathcal{A}}_t = \bigcup_{h' \in \mathcal{H}} \pi_{\theta_{h'}}(\mathcal{O}_t^{h'})$
17: Calculate $\tilde{\mathbb{S}}_t^{h}$ according to (32)
18: $\mathcal{L}(\theta_h) = \mathbb{E}_{\boldsymbol{\mathcal{D}}}[\min(Q_{\psi_h^i}(\mathcal{S}_t)) + \omega_h \tilde{\mathbb{S}}_t^{h}]|_{i=1,2}$
19: $\theta_h = \text{Adam}\big(\nabla_{\theta_h}(-\mathcal{L}(\theta_h))\big)$
20: $\hat{\theta}_h = \mu \hat{\theta}_h + (1-\mu)\theta_h$, $\hat{\psi}_h^i = \mu \hat{\psi}_h^i + (1-\mu)\psi_h^i|_{i=1,2}$
21: **if** $t' \% \lambda_{\mathbb{S}} == 0$ **then**
22: **for all** $h \in \mathcal{H}$ **do**
23: Sample $\boldsymbol{\mathcal{D}}$ and extract $\mathcal{O}_t^h$, $\tilde{a}_h(t) = \pi_{\theta_h}(\mathcal{O}_t^h)$
24: Calculate $\mathcal{L}(\omega_h)$ according to (34)
25: $\omega_h = \text{Adam}(\nabla_{\omega_h}(-\mathcal{L}(\omega_h)))$

11–12). When updating an actor, the entropy of the distributed task allocation actions is calculated (line 17). Then, the loss value integrated with the action entropy of massive users is calculated (line 18), and the actor is updated (line 19). Target networks are updated by the soft update method (line 20), where μ is the soft-update coefficient. This paper also employs the delayed update method to update ω_h but sets a different update frequency $\lambda_{\mathbb{S}}$. We find by experiment that updating ω_h faster than the actor benefits convergence. Thus, we let $\lambda_{\mathbb{S}} < \lambda$, and the learning rate for updating ω_h greater than that used to update other policies. When updating ω_h, the extracted local states are processed by respective actors to generate new actions (line 23). Then, ω_h is updated by processing the loss value calculated from these generated actions (lines 24–25).

F. Complexity analysis

The policy training is performed on the RCS with sufficient computing power in an off-policy style. Therefore, we should care more about the complexity of policy execution on ESs. The inference is a forward propagation process conducted by a multi-layer perceptron (MLP). The matrix computation of MLP layers determines the inference complexity [17], [47]–[49]. Besides, there is a *softmax* operation on the output in DJTCRA policy, whose complexity is equal to the number of processed items. Therefore, the inference

Algorithm 2 Agent execution and experience collection

Input: EC system, DJTCRA policies, replay buffer

1: **for** $t' = 1, 2, 3, \cdots$ **do**
2: Each ES h observes its local state $\mathcal{O}_{t'}^h$
3: Each ES h implements an action $a_{t'}^h = \pi_{\theta_h}(\mathcal{O}_{t'}^h)$
4: Each ES h obtains information of its PA users $\mathcal{U}_{t'}^h$
5: Every ES allocates the resource to its PA users
6: Each ES h obtains $r_{t'}^h$ and $\mathcal{O}_{t'+1}^h$
7: Every ES uploads $(\mathcal{O}_{t'}^h, a_{t'}^h, r_{t'}^h, \mathcal{O}_{t'+1}^h)$
8: RCS calculates $\mathcal{S}_{t'}$, $\mathcal{A}_{t'}, \mathcal{R}_{t'}$ and $\mathcal{S}_{t'+1}$
9: RCS adds $(\mathcal{S}_{t'}, \mathcal{A}_{t'}, \mathcal{R}_{t'}, \mathcal{S}_{t'+1})$ to the replay buffer

complexity of a DJTCRA policy performed on each ES is $\mathcal{O}(\overline{s}k_0 + \sum_{i=0}^{\hat{h}-1} k_i k_{i+1} + k_{\hat{h}}\overline{a} + \overline{a})$, where $\overline{s}$ is the input state size (i.e., $|\mathcal{U}| \times 9$), k_0 is the output size of the input layer. $\hat{h}$ is the number of hidden layers, and k_i is the hidden size of the i^{th} hidden layer. $\overline{a}$ is the size of the output action (i.e., $|\mathcal{U}| \times |\mathcal{H}|$).

VI. Performance Evaluation

A. Simulation Setup

We conduct extensive simulations using PyTorch 1.9 and the sigmoidal programming optimizer with the support of PyJulia [36], [50]. We deploy 7 RAN nodes, and the computing resource capacities $\mathcal{P}^h$ of ESs associated with each RAN are different and set to $[60, 60, 80, 90, 100, 120, 160]$ GHz. The communication rate of links between RANs and a user's processing performance variation factors among ESs (i.e., η_u^h) are randomly selected from the configured intervals and fixed. To imitate the unbalanced speed of ESs' computing, we randomly generate the number of users on each ES whose tasks can obtain the maximum processing speed (i.e., minimum η_u^h). Then, we randomly sample users for each ES according to the generated number and set the corresponding η_u^h as the minimum value by swapping operations. In each training episode, the user positions (connected RAN, distance to the connected RAN), wireless communication noise, bandwidth, computing requirements, and QoE-QoS correlations are all randomly generated from configured intervals and distributions. The correlation coefficients (i.e., α_u and β_u) are generated following normal distributions. Other parameters are randomly generated following uniform distributions [23]. TABLE III details parameter configurations [32]. The maximum QoE a user can achieve (L_u) is homogeneously set to 1, reflecting the QoE satisfaction ratio. In addition, the communication rate and propagation delay between RAN nodes and the RCS are separately set to 100 Mbps and 200 ms, respectively. Besides, we set the task processing speed on RCS to 20 GHz. In addition, parameters related to DJTCRA policy networks and policy training are detailed in TABLE IV.

We evaluate and compare the proposed approach with MADDPG which is widely employed in previous works [10], [16], [19], [43]. Besides, our framework cannot directly adopt the value-based methods since generating and comparing the Q-values of all candidate actions for more than hundreds of users is impossible. Therefore, we compare the value-based MADRL methods that enable each user to decide their task

TABLE III
SIMULATION PARAMETERS

Parameter	Value	Parameter	Value
$\mathcal{B}_{i,j}$	[8, 20]×10 Mbps	$\mathbb{H}_{UT}$	1.5 m
$\mathcal{B}_u$	[2, 7]×8 MHz	h	5 m
$\mathcal{N}_u$	[-120, -80] dbm	$\mathbb{H}_{BS}$	35 m
α_u	$N(15, 4^2)/1000$	f_c	5 GHz
η_u^h	[0.5, 8]	$\mathcal{I}_u$	20 dbm
κ_u	[228, 328] cycles/bit	Υ_{2d}	[30, 100] m
V_u^R	[10, 20] KB	$\delta_{i,j}$	[10, 20] ms

TABLE IV
CONFIGURATIONS RELATED TO MADRL APPROACHES

Configuration	Value
The input and output size of hidden layers	256
The depths of hidden layers	3
The activation function between hidden layers	ReLU
The delayed update frequency $\lambda, \lambda_{\mathbb{S}}$	15, 5
The soft update coefficient μ	0.995
The learning rate of policy networks	1e-4
The learning rate of max entropy coefficient ω_h	1e-3

allocation, as proposed in many previous works [8], [17], [51]. The comparison benchmarks include:

1) **MADDPG:** Each ES allocates the tasks of its CA users as described in our work. However, we train the policies using MADDPG, in which the training process is done without integrating the entropy of massive users' task allocation actions [18], [19].
2) **VDN-L:** Each user determines its task allocation based on local state observation (i.e., state of the user). The policies are trained by the VDN algorithm, in which the policies of all users are updated simultaneously by utilizing the accumulating Q values of all policies. More details can be seen from [52].
3) **VDN-G:** Each user determines its task allocation with the input of the global state (i.e., system state). We set each user to obtain the state of other users from a virtual central unity or by interactions, as assumed in some previous work [15], [29].
4) **IDQN:** Each user maintains and trains a policy by DQN independently, and the policy can determine its task allocation with local state observation [8].
5) **GA-Aware**: Greedy algorithm that assumes that each user knows the ES that can process its task request with fast speed under given resources and select this ES to process its tasks, i.e., $h^* = \arg\min\{\eta_u^h\}$.
6) **DAES:** Each user's tasks are allocated to the directly associated ES rather than a remote ES.

We set the percentile of greedy selecting a random action for exploration in VDN and IDQN as 0.15. We evaluate approaches using the following metrics:

1) **System QoE:** Sum QoE of all users after implementing a DJTCRA action, which is the primary objective and can be calculated by (9), (11), and (12).
2) **Average service latency:** The service latency of each user can be calculated based on (9). We average the service latency of all users for visualization.
3) **Users distribution:** The number of users classified by QoE, including QoE-satisfied users (SU), QoE-disturbed users (DU), and users with extremely disturbed QoE (GU). The QoE thresholds for classifying SU, DU, and GU are separately set to 0.9 and 0.1. We also count the users whose tasks are offloaded to RCS (CU).

B. Simulation Results

Fig. 6 compares the convergence of MADRL-based approaches over the training process and compares them with traditional approaches. The results of every 50 training episodes are moving-averaged and displayed. From Fig. 6, we can find that the performance of IDQN is barely better after convergence and is even worse than DAES, which suggests that policies do not learn meaningful information. Compared with IDQN and DAES, VDN can help achieve better performance in average system QoE and latency. Besides, it can reduce the number of GUs as compared with DAES. Moreover, despite the acceleration of convergence, even with the global system state of all users, VDN-G cannot achieve higher performance than that only with local state observation (i.e., VDN-L). However, policies trained using IDQN and VDN for users perform worse than those using MADDPG and MAMaxEnt for ESs because inferring and adapting to a mass of other users' policies updated during the training process is challenging. When MADDPG is employed, the QoE and latency are significantly superior to those utilizing IDQN, DAES, and VDN. Besides, there are considerably more SUs and DUs but fewer GUs and CUs in the system with MADDPG. The greedy algorithm GA-Aware achieves slightly better performance (i.e., higher system QoE and lower latency) than MADDPG. The reason is that when allocating tasks to the ES with the fastest processing speed for it, the QoE

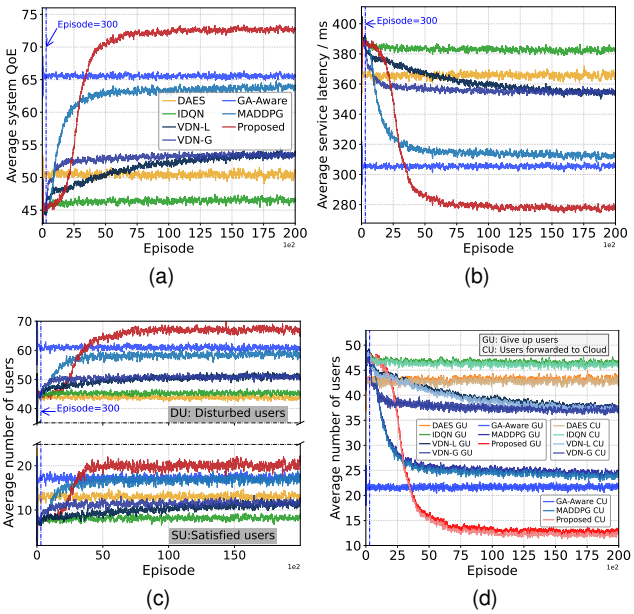


Fig. 6. Training process. $|\mathcal{U}| = 100$, $\gamma = 0.1$, $\beta_u \sim N(350, 10^2)$ ms, $V_u^T \sim U(200, 800)$ KB. (a) Average system QoE. (b) Average service latency. (c)-(d) Average number of SUs, DUs, GUs, and CUs.

of the user can be satisfied with fewer resources, making more resources available to please more users, especially those requiring fewer resources. Then, the service latency decreased, and system QoE increased because of the lower task processing latency. Compared with MADDPG and GA-Aware, our proposed approach further improves system QoE and reduces average service latency. Besides, there are further increases in SUs and DUs and decreases in GUs and CUs. Fig. 6d reveals that the number of CUs is slightly fewer than that of GUs and almost the same. This phenomenon demonstrates that EC system resource limitations primarily cause severe QoE disruption. We can also find that the proposed approach outperforms MADDPG regarding convergence speed and optimal performance. The performances of MADDPG continue to increase extremely slowly after about 3000 training episodes. However, the proposed approach achieves optimal performance and remains stable after 8000 training episodes. Moreover, the performance improvement of our approach is slower than others at the initial training stage because the initial max entropy coefficients are not optimal, resulting in policies taking more random actions.

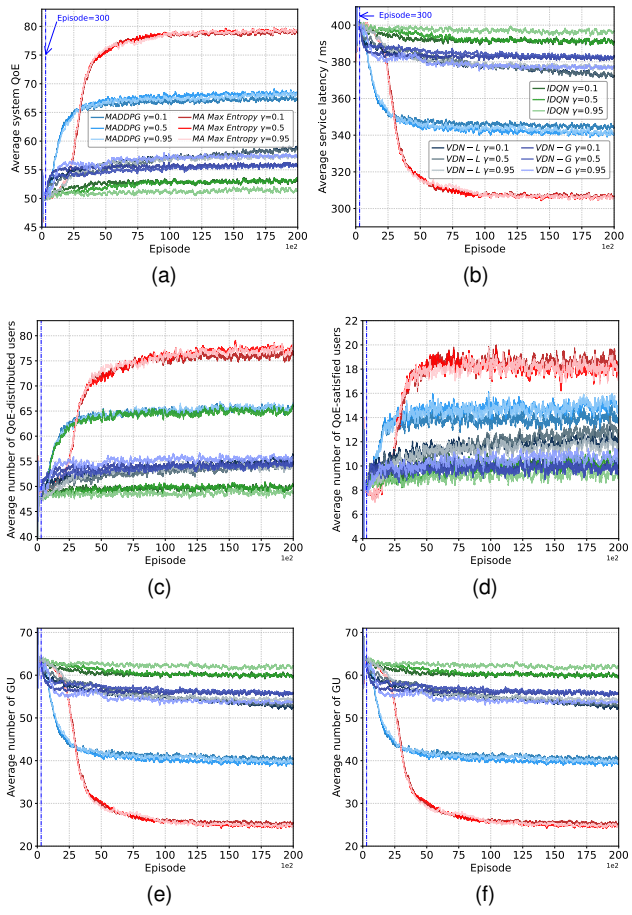


Fig. 7. Inpact of discount factors. $|\mathcal{U}|=120$, $\beta_u \sim N(350, 10^2)$ ms, $V_u^T \sim U(200, 800)$ KB. (a) Average system QoE. (b) Average service latency. (c)-(f) Average number of DUs, SUs, GUs, and CUs.

Fig. 7 evaluates the convergence of MADRL-based approaches under different discount factors. We moving-average the results for every 100 episodes. We can find that the results of all algorithms present apparent convergence, although there is a slight difference in the training process. Under different discount factors, each of them can achieve similar convergent performance. Furthermore, policies trained for ESs using the proposed approach and MADDPG outperform policies for users using IDQN and VDN in terms of performance after convergence and stability. In addition, our proposed approach achieves optimal performance and is more stable, as other approaches expose an apparent performance difference after convergence under different discount factors.

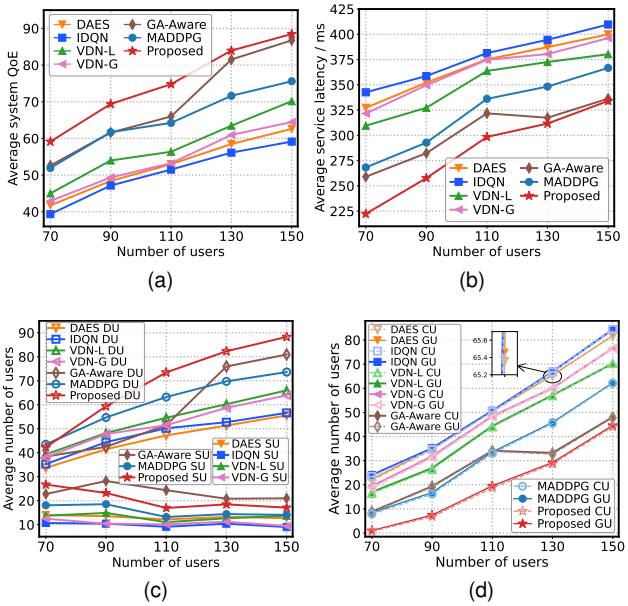


Fig. 8. Impact of the number of users. $\beta_u \sim N(350, 10^2)$ ms, $\gamma = 0.1$, $V_u^T \sim U(200, 800)$ KB. (a) Average system QoE. (b) Average service latency. (c) Average number of SUs and DUs. (d) Average number of GUs and CUs.

Fig. 8 compares the performance under different numbers of users. We average the results of the last 2000 episodes after 20000 training episodes for each simulation. Regardless of the approach, the average system QoE and service latency will increase with the number of users. The reason is that more users with lower computing requirements can be served, which increases the sum of QoE since users' QoE is non-negative. The number of SUs decreases slightly, but the number of DUs significantly increases as the number of users increases. The potential reason is that the reward function is the system QoE without considering the number of SUs. Therefore, the computing resources are separated to improve the QoE of users whose QoE is in the disturbed area, which can be significantly increased with more computing resources. Besides, as shown in Fig. 8(a), except for when there are 70 users, the number of SU obtained by GA-Aware is slightly higher than that by the proposed approach. The reason is that GA-Aware always selects the ES that can maximize the task processing speed, which increases the opportunity for users who require fewer resources to be satisfied. When an ES only provides service for several users whose tasks can processed on it with the fastest speed, its resources are sufficient to satisfy the QoE of all these users. Consequently, compared with our proposed approach, the number of SUs increased slightly. However,

it leads to inefficient resource utilization. Thus, the overall QoE has decreased, and the number of GUs is slightly higher. When there are 70 users, the resources of ESs are sufficient to provide service to almost all users. The proposed method can better utilize ESs' resources, resulting in a little more SU, significantly more DUs, and almost no DUs (Fig. 8(d)). Fig. 8(d) reveals that the number of CUs and GUs increases as the number of users increases. The reason is that more users have to be reallocated to the RCS due to the resource limitations of ESs, resulting in considerable service latency and an extremely disturbed QoE. As a result, the average service latency increased (Fig. 8b). Besides, the proposed approach can minimize the number of CUs and GUs. Fig. 8 reveals that IDQN and VDN perform similarly to DAES, while MADDPG, GA-Aware, and our proposed approaches outperform them. We can also find that when VDN is allowed to gather the global states (VDN-G), it performs even a little worse than when only the local states are available (VDN-L). The possible reason is that the VDN-based approaches cannot enable massive policies in our problem to learn efficient knowledge, which can be concluded from the above results. Then, the global system state increases the complexity of the input data that the policy needs to process and the difficulties of policy training. Then, the reliability of trained policies decreases. Besides, GA-Aware achieves better than MADDPG, especially when there are more than 110 users. The reason is that the percentile of users requiring fewer resources to satisfy QoE has increased, and the resources of each ES will be fully consumed as the increase in computing load arrives at each ES. Additionally, the number of users served by ESs obtained by GA-Aware is comparable to that obtained using our proposed approach, and these users have similar requirements. Consequently, GA-Aware obtains similar performance as our proposed approach. However, our proposed approaches can still outperform all other benchmarks. Besides, we can find that when the ESs' resources are adequate to support users, i.e., only a few users need to be allocated to RCS, the proposed approach outperforms others.

The correlation between QoE and QoS determines the resources required to enable users to achieve different levels of QoE. Therefore, Fig. 9 compares methods under different QoE and QoS correlation coefficients, i.e., the average middle point value. We set different mean values while maintaining the same variance in generating the β_u of users. In this part, we use β_u to represent the average value of the middle point for easy expression. We can find that the average system QoE increases with the increase of β_u regardless of the employed approach. When fixing allocated resources, users can tolerate higher service latency under a higher β_u and obtain a better QoE. Under higher β_u conditions, users can achieve the same QoE as in lower β_u conditions with fewer resources. Then, some users may be allocated fewer computing resources, and the freed resources are allocated to other users to maximize the system QoE. As a result, some users' task processing can be moved from the RCS to ESs, reducing their service latency. However, reducing the resources allocated to users increases their service latency. Thus, there are decreases in GUs and CUs as the β_u increases (Fig. 9(d)), and the average service latency only experiences a slight increase (Fig. 9(c)). Besides, users' QoE can be satisfied with fewer resources under a higher β_u condition, reducing the resource required for satisfying a user and increasing the probability of satisfying the QoE of more users. Thus, Fig. 9(c) exhibits that SUs increase as the β_u increases. The increase in SUs reduces the percentile of DUs and GUs. Thus, DUs decrease as β_u increases, while the decrease in DUs is fewer than the increase in SUs. Fig. 9 also reveals that IDQN can hardly learn effective policy, and VDN-based approaches can slightly improve performance compared to DAES and IDQN. GA-Aware obtains slightly better results than MADDPG. Our proposed method outperforms other benchmarks in terms of QoE and latency. Besides, our proposed method can minimize the number of DUs and CUs, although GA-Aware results in a few more SUs. We can conclude that our proposed approach can improve resource efficiency compared with benchmarks.

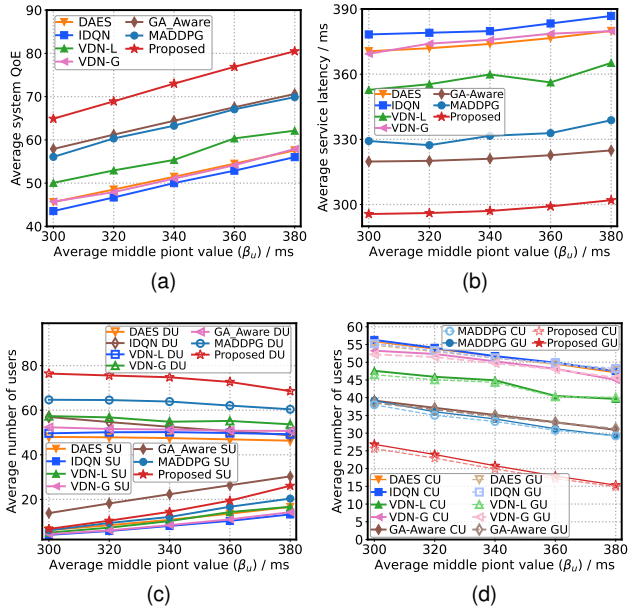


Fig. 9. Impact of the average middle point value. $|\mathcal{U}| = 110$, $\gamma = 0.1$, $V_u^T \sim U(200, 800)$ KB. (a) Average system QoE. (b) Average service latency. (c) Average number of SUs and DUs. (d) Average number of GUs and CUs.

Fig. 10 evaluates the impact of maximum task volume (MTV). First, we can observe that VDN and IDQN achieve similar results as DAES. VDN-based approaches achieve slightly better performance in QoE and latency than DAES, but IDQN results in the lowest performance. MADDPG, GA-Aware, and our proposed approaches outperform VDN, IDQN, and DAES. Besides, the proposed approach achieves the maximum system QoE, minimum latency, and minimum number of CUs. The average system QoE decreases, and the average service latency increases as the MTV increases because increased task volume increases wireless communication, forwarding, and task processing latency. Besides, more resources are required by users' task processing, resulting in resource strain for ESs. Then, there are increases in CUs (Fig. 10(d)) and service latency, decreasing the average system QoE. As shown in Fig. 10(c), the number of SUs and DUs decreases as MTV increases since more resources are required to

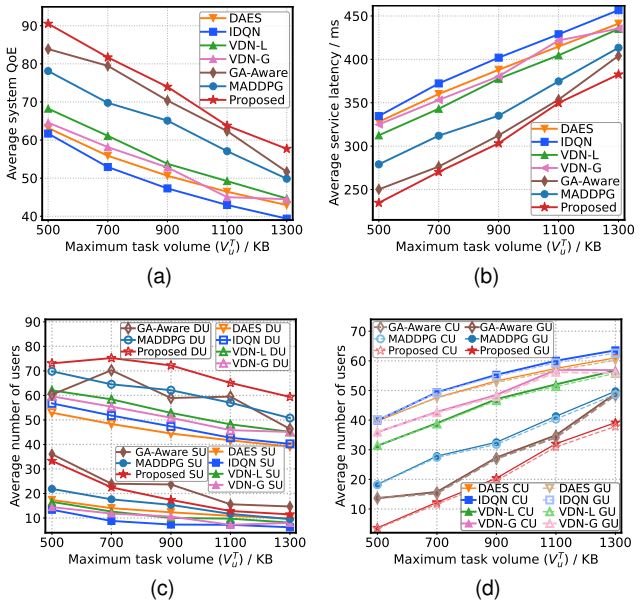


Fig. 10. Impact of the maximum computing task volume. $|\mathcal{U}|=110$, $\gamma=0.1$ $\beta_u \sim N(350, 10^2)$ ms. (a) Average system QoE. (b) Average service latency. (c) Average number of SUs and DUs. (d) Average number of GUs and CUs.

satisfy QoE. More users are allocated to the RCS and become GUs. The number of DUs that resulted from the proposed approach and GA-Aware increased when MTV increased from 500 KB to 700 KB. When the MVT is small, users require fewer resources to satisfy their QoS. In such conditions, the resources of ESs are sufficient, and there is a high probability that the QoE of a user is satisfied when employing the GA-Aware approach and our proposed approach. The reason is that their targets are to reduce the computing resource consumption of each user and to utilize limited resources more efficiently through optimal task allocation, respectively. However, some users' QoEs rapidly declined and fell into the QoE-disturbed area when the task volume increased because resources became insufficient. Consequently, the number of SUs significantly decreases, but the number of DUs slightly increases on the left side of Fig. 10(c) since more users' QoE falls into the QoE-disturbed area. Moreover, the GA-Aware approach exhibits more instability than others, as its performance heavily relies on the distribution of users and their requirements, rendering it unreliable, especially in certain special cases. For instance, a super ES can deliver maximum processing speed for all users in a random system. The edge system can support nearly all users if it employs our recommended approach when the MTV is 500 KB.

VII. Conclusion

This paper studied a distributed EC system with massive users. Then, the DJTCRA problem of maximizing over QoE is set up, considering both resource efficiency and the quantitative correlation between QoE and QoS. The restriction that each ES can only obtain the state of real-time associated users and allocate its limited resources to served users by itself has been comprehensively considered. According to the work procedures and the restrictions on state observation and resource allocation at different stages, we decomposed the DJTCRA problem into a distributed task allocation problem and a resource allocation problem. To address the problem, this paper developed a two-step MADRL-based approach after formulating it as a POMDP. This solution first proposes a policy-based MADRL method to realize the massive users' distributed task allocation with vast discrete action spaces. Besides, considering the challenges of policy training for numerous agents, this paper selects ESs as the agents. Then, we further integrated it with the action entropy of enormous users' task allocation to enhance exploration. Besides, we proved that resource allocation for maximizing QoE is a problem of maximizing a sum of sigmoids, and we leveraged sigmoidal programming to solve it. Extensive simulation results demonstrated that the proposed approach can improve QoE and reduce service latency compared with benchmarks. Besides, the proposed approach outperforms classic MADDPG in training efficiency and performance after convergence.

Appendix
Proof of the *Proposition 1*

Define a function $f(x) = h(g(x))$, where x is the variable and $x > 0$. Meanwhile, $g(x) = C/x$ and

$$h\left(z\right) = \frac{1}{\left(1 + e^{\alpha(M-\beta+z)}\right)}, \tag{35}$$

where α, M, β, and C are all constant values greater than 0. Thus, the derivative of $h(z)$ to z is always negative, i.e.,

$$h'(z) = -\frac{\alpha e^{\alpha(M-\beta+z)}}{(e^{\alpha(M-\beta+z)}+1)^2} < 0. \tag{36}$$

In addition, the second derivative of $h(z)$ is

$$h''(z) = \frac{\alpha^2 e^{\alpha(M-\beta+z)}(e^{\alpha(M-\beta+z)}-1)}{(e^{\alpha(M-\beta+z)}+1)^3}. \tag{37}$$

The second-order derivative of $f(x)$ to variable x is

$$f''(x) = h''(g(x))\underbrace{g'(x)^2}_{>0} + \underbrace{h'(g(x))}_{<0}\underbrace{g''(x)}_{>0}. \tag{38}$$

It's obvious that $f''(x)$ must be negative when $h''(g(x))$ is negative. From (37), we have $h''(z) < 0$ when $(M-\beta+z) < 0$. Then, since $x > 0$, when $(\beta - M) > 0$ we have

$$h''(g(x)) < 0, \forall x > \frac{C}{\beta - M}, \tag{39}$$

In addition, from (36), (37), we can express (38) as (40), where $z = \frac{C}{x}$. As we have $x > 0$, the positiveness and negativity of $f''(x)$ depend on the numerator of (40). By omitting a positive constant value C, we express the numerator as

$$\underbrace{e^{\alpha\left(M-\beta+\frac{C}{x}\right)}}_{>1}\left(\left(\alpha^2-2x\alpha\right)\underbrace{e^{\alpha(M-\beta+\frac{C}{x})}}_{>1}-\underbrace{\left(\alpha^2+2x\alpha\right)}_{>0}\right). \tag{41}$$

Then, as $\alpha > 0$ and $C/x > 0$, we have $e^{\alpha(M-\beta+C/x)} > 1$ and $2x\alpha > 0$ when $(\beta - M) \leq 0$. Then, when $(\beta - M) \leq 0$ and $x > \alpha/2$, we can obviously find that

$$\left(\left(\alpha^2 - 2x\alpha\right)e^{\alpha(M-\beta+C/x)} - \left(\alpha^2 + 2x\alpha\right)\right) < 0 \tag{42}$$

$$
\begin{aligned}
f''(x) &= \frac{\alpha^2 e^{\alpha(M-\beta+z)}\left(e^{\alpha(M-\beta+z)}-1\right)}{\left(e^{\alpha(M-\beta+z)}+1\right)^3}\frac{C}{x^4} + \frac{-\alpha e^{\alpha(M-\beta+z)}}{\left(e^{\alpha(M-\beta+z)}+1\right)^2}\frac{2C}{x^3} \\
&= \frac{\alpha^2 e^{\alpha(M-\beta+z)}\left(e^{\alpha(M-\beta+z)}-1\right) - 2x\alpha e^{\alpha(M-\beta+z)}\left(e^{\alpha(M-\beta+z)}+1\right)}{x^4\left(e^{\alpha(M-\beta+z)}+1\right)^3}C
\end{aligned} \tag{40}
$$

Thus, when $(\beta - M) \leq 0$, when have

$$f''(x) < 0, \forall x > \frac{\alpha}{2}. \tag{43}$$

According to the above analysis, we can conclude that there always exists a value ϵ, making $f''(x) < 0, \forall x > \epsilon$, i.e, $f(x)$ is up concave when $x > \epsilon$.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

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

[3] Y. Zhou, L. Liu, L. Wang, N. Hui, X. Cui, J. Wu, Y. Peng, Y. Qi, and C. Xing, "Service-aware 6g: An intelligent and open network based on the convergence of communication, computing and caching," *Digital Communications and Networks*, vol. 6, no. 3, pp. 253–260, 2020.

[4] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[5] D. Wu, R. Bao, Z. Li, H. Wang, H. Zhang, and R. Wang, "Edge-cloud collaboration enabled video service enhancement: A hybrid human-artificial intelligence scheme," *IEEE Transactions on Multimedia*, vol. 23, pp. 2208–2221, 2021.

[6] T. M. Ho and K.-K. Nguyen, "Joint server selection, cooperative offloading and handover in multi-access edge computing wireless network: A deep reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2421–2435, 2022.

[7] D. Wu, Q. Liu, H. Wang, D. Wu, and R. Wang, "Socially aware energy-efficient mobile edge collaboration for video distribution," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2197–2209, 2017.

[8] Y. Zhang, B. Di, Z. Zheng, J. Lin, and L. Song, "Distributed multi-cloud multi-access edge computing by multi-agent reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2565–2578, 2021.

[9] J. Heydari, V. Ganapathy, and M. Shah, "Dynamic task offloading in multi-agent mobile edge computing networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[10] Z. Cheng, M. Min, Z. Gao, and L. Huang, "Joint task offloading and resource allocation for mobile edge computing in ultra-dense network," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[11] H. Lu, C. Gu, F. Luo, W. Ding, S. Zheng, and Y. Shen, "Optimization of task offloading strategy for mobile edge computing based on multi-agent deep reinforcement learning," *IEEE Access*, vol. 8, pp. 202 573–202 584, 2020.

[12] Y. Gong, J. Wang, and H. Yao, "Distributed multi-agent empowered resource allocation in deep edge networks," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, 2021, pp. 974–979.

[13] X. Liu, J. Yu, Z. Feng, and Y. Gao, "Multi-agent reinforcement learning for resource allocation in iot networks with edge computing," *China Communications*, vol. 17, no. 9, pp. 220–236, 2020.

[14] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, "Multi-agent drl for task offloading and resource allocation in multi-uav enabled iot edge network," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4531–4547, 2021.

[15] C. Hu, J. Li, H. Shi, B. Ning, and Q. Gu, "Decentralized offloading strategies based on reinforcement learning for multi-access edge computing," *Information*, vol. 12, no. 9, 2021.

[16] L. Ma, H. Shi, J. Li, and K.-S. Hwang, "A multi-agent reinforcement learning based offloading strategy for multi-access edge computing," in *2021 International Automatic Control Conference (CACS)*, 2021, pp. 1–5.

[17] L. Chen and J. Xu, "Seek common while shelving differences: Orchestrating deep neural networks for edge service provisioning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 251–264, 2021.

[18] H. Lin, W. Hou, H. Wen, W. Lei, S. Wu, and Z. Chen, "Maddpg-based task offloading and resource management for edge system," in *The 2nd International Conference on Computing and Data Science*, ser. CONF-CDS 2021. New York, NY, USA: Association for Computing Machinery, 2021.

[19] 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 of Things Journal*, vol. 8, no. 22, pp. 16 256–16 268, 2021.

[20] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[21] B. Gu, X. Yang, Z. Lin, W. Hu, M. Alazab, and R. Kharel, "Multiagent actor-critic network-based incentive mechanism for mobile crowdsensing in industrial systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6182–6191, 2021.

[22] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.

[23] S. Li, J. Huang, J. Hu, and B. Cheng, "Qoe-deer: A qoe-aware decentralized resource allocation scheme for edge computing," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2021.

[24] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Qoe-aware user allocation in edge computing systems with dynamic qos," *Future Generation Computer Systems*, vol. 112, pp. 684–694, 2020.

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

[26] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[27] 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 Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1199–1226, 2023.

[28] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A distributed and context-aware task assignment mechanism for collaborative mobile edge computing," *Sensors*, vol. 18, no. 8, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2423

[29] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 85–97, 2019.

[30] X. Wang, J. Ye, and J. C. Lui, "Decentralized scheduling and dynamic pricing for edge computing: A mean field game approach," *IEEE/ACM Transactions on Networking*, vol. 31, no. 3, pp. 965–978, 2023.

[31] J. Yang, Q. Yuan, S. Chen, H. He, X. Jiang, and X. Tan, "Cooperative task offloading for mobile edge computing based on multi-agent deep reinforcement learning," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.

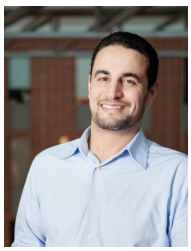
[32] ETSI, "Lte;5g; study on channel model for frequency spectrum above 6 ghz (3gpp tr 38.900 version 14.2.0 release 14)," European Telecommunications Standards Institute,3rd Generation Partnership Project (3GPP), Technical Report (TR) 138 900, 06 2017, version

14.2.0. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/138900_138999/138900/14.02.00_60/tr_138900v140200p.pdf
[33] P. Lai, Q. He, G. Cui, F. Chen, M. Abdelrazek, J. Grundy, J. Hosking, and Y. Yang, "Quality of experience-aware user allocation in edge computing systems: A potential game," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 2020, pp. 223–233.
[34] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge, UK: Cambridge university press, 2004.
[35] M. Hemmati, B. McCormick, and S. Shirmohammadi, "Qoe-aware bandwidth allocation for video traffic using sigmoidal programming," *IEEE MultiMedia*, vol. 24, no. 4, pp. 80–90, 2017.
[36] M. Udell and S. Boyd, "Maximizing a sum of sigmoids," May 2015. [Online]. Available: https://people.orie.cornell.edu/mru8/doc/max_sum_sigmoids.pdf
[37] Y. Chen, Y. Sun, C. Wang, and T. Taleb, "Dynamic task allocation and service migration in edge-cloud iot system based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 742–16 757, 2022.
[38] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 1126–1135.
[39] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.
[40] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *35th International conference on machine learning*. PMLR, 2018, pp. 1587–1596.
[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT press, 2018.
[42] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, California,USA, 2017, pp. 6382–6393.
[43] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. M. Leung, "Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 154–169, 2021.
[44] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.
[45] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *arXiv preprint*, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1801.01290
[46] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint*, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1910.07207
[47] Q. Tang, R. Xie, F. R. Yu, T. Chen, R. Zhang, T. Huang, and Y. Liu, "Distributed task scheduling in serverless edge computing networks for the internet of things: A learning approach," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19 634–19 648, 2022.
[48] Y. Liu, H. Wang, M. Peng, J. Guan, and Y. Wang, "An incentive mechanism for privacy-preserving crowdsensing via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8616–8631, 2021.
[49] D. Wu, H. Shi, H. Wang, R. Wang, and H. Fang, "A feature-based learning system for internet of things applications," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1928–1937, 2019.
[50] "Pyjulia," The Julia and IPython development teams, 2019. [Online]. Available: https://pyjulia.readthedocs.io/en/latest/index.html
[51] A. Suzuki, M. Kobayashi, and E. Oki, "Multi-agent deep reinforcement learning for cooperative computing offloading and route optimization in multi cloud-edge networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.
[52] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 2085–2087.

Yan Chen received the B.E. degree in Information Engineering and the Ph.D. degree in Information & Communication Engineering from China University of Mining and Technology, China, in 2016 and 2022. He is currently a Postdoctoral Researcher with Zhejiang Lab, China. He was also a visiting Ph.D. student at Aalto University, Finland. His research interests include Edge Computing, Internet of Things, and Wireless networks.

Yanjing Sun has been a Professor in the School of Information and Control Engineering at the China University of Mining and Technology since July 2012. He received the Ph.D. degree in Information and Communication Engineering from the China University of Mining and Technology in 2008. His current research interests include wireless communication, wireless sensor networks, Cyber-physical systems, and so on.

Hao Yu received the B.E. and Ph.D. degree in communication engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2015 and 2020. He was also a Joint-Supervised Ph.D. Student with the Politecnico di Milano, Milano, Italy. He is currently a Postdoctoral Researcher with the Center of Wireless Communications, University of Oulu, Finland. His research interests include network automation, SDN/NFV, time-sensitive networks, and deterministic networking.



Tarik Taleb (Senior Member, IEEE) received the B.E. degree (with distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a Full Professor at Ruhr University Bochum, Germany. He was a Professor with the Center of Wireless Communications, University of Oulu, Oulu, Finland. He is the founder of ICTFICIAL Oy, and the founder and the Director of the MOSA!C Lab, Espoo, Finland. From October 2014 to December 2021, he was an Associate Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. Prior to that, he was working as a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. Before joining NEC and till March 2009, he worked as an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, in a lab fully funded by KDDI. From 2005 to 2006, he was a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. Taleb has been directly engaged in the development and standardization of the Evolved Packet System as a member of the 3GPP System Architecture Working Group. His current research interests include AI-based network management, architectural enhancements to mobile core networks, network softwarization and slicing, mobile cloud networking, network function virtualization, software-defined networking, software-defined security, and mobile multimedia streaming.