Transformer-based Distributed Task Offloading and Resource Management in Cloud-Edge Computing Networks

Mingqi Han*, Xinghua Sun*, Xijun Wang†, Wen Zhan*, Xiang Chen†

***Abstract*—ndustrial Cyber-Physical Systems (ICPS) have emerged as a critical component in the industrial domain. To facilitate seamless collaboration among massive devices, cloud-edge computing architectures have emerged as a key enabler for ICPS, leveraging distributed intelligence to orchestrate devices and computational tasks. In cloud-edge computing, efficient task offloading and resource management are essential for optimizing task performance and reducing energy costs. However, conventional centralized resource management strategies struggle to satisfy the real-time, adaptability, and performance demands of dynamic ICPS systems.ndustrial Cyber-Physical Systems (ICPS) have emerged as a critical component in the industrial domain. To facilitate seamless collaboration among massive devices, cloud-edge computing architectures have emerged as a key enabler for ICPS, leveraging distributed intelligence to orchestrate devices and computational tasks. In cloud-edge computing, efficient task offloading and resource management are essential for optimizing task performance and reducing energy costs. However, conventional centralized resource management strategies struggle to satisfy the real-time, adaptability, and performance demands of dynamic ICPS systems.I In this paper, we propose the Distributed Transformer-based Actor-Critic (DTAC) algorithm to jointly determine task offloading and resource management decisions in cloud-edge computing networks, particularly for delay-sensitive applications in ICPS. The DTAC algorithm integrates the powerful transformer model with the popular actor-critic architecture to address the challenge of a hybrid high-dimensional action space. We first train a centralized model to learn coordination among user equipments (UEs) and then introduce a decentralized transfer learning (TL) approach to efficiently adapt the centralized model into the DTAC framework. Using the DTAC model, each UE can independently manage its local resources based solely on local information, avoiding the significant signaling overhead inherent in centralized approaches. Simulation results demonstrate that DTAC not only outperforms other MARL and TL schemes in both small- and large-scale scenarios, but also exhibits strong generalization capabilities in inexperienced settings. Furthermore, DTAC and decentralized TL approaches significantly reduce training costs by 73% compared to other methods, making them more practical for ICPS deployment.**

***Index Terms*—Cloud-edge Computing, Deep Reinforcement Learning, Distributed Resource Management, Transfer Learning**

I. Introduction

With the rapid development of communication technology and the internet of thing (IoT), Cyber-Physical Systems (CPS) have emerged to seamlessly integrate the cyber world with the physical world by monitoring and controlling physical entities through intelligent computing and communication technologies [1]. Motivated by the CPS, the intelligent devices have widely deployed in manufacturing sectors, prompting a fundamental paradigm shift from traditional industrial settings to the Industrial Cyber-Physical Systems (ICPS) [1]–[7].

The ICPS enable real-time monitoring, coordination, and integration of massive physical devices, promising a framework to provide sophisticated functions. In ICPS, the primary challenge then lies in the management of massive heterogeneous interconnected devices in industrial environments. Towards this end, cloud-to-edge-based ICPS has been proposed to enhance connectivity, networked computing, and intelligent control by leveraging existing cloud/edge technologies [6]–[8]. Within the cloud-to-edge-based ICPS, devices can benefit from significantly enhanced computational and communication capabilities at reduced costs and latency. Within this framework, cloud-edge computing plays a critical role in the intelligent control of industrial devices [7]–[11]. Leveraging cloud-edge computing, resource-constrained industrial devices can access additional computational resources from other devices under the management of a cloud server. This collaborative paradigm reduces task execution delays, improves system-wide performance, and lowers energy consumption [12], [13].

In cloud-edge computing networks, task offloading and resource management are essential to selectively offload tasks to edge servers and allocate appropriate communication and computation resources for each task [14]–[23]. By jointly optimizing offloading decisions and the allocation of communication and computation resources, both execution time and energy consumption for user equipment (UEs) can be significantly reduced, which is highly desirable. However, such joint resource management inherently forms a challenging mixed-integer nonlinear programming (MINLP) problem, involving an integer component for computation placement and com-

The work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB2904100, in part by the National Natural Science Foundation of China under Grant 62271513, in part by Guangdong S&T Program under Grant 2024B0101010001, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515012015, in part by the Fundamental Research Funds for the Central Universities, Sun Yat-sen University, under Grant 24qnpy204, and in part by The Shenzhen Science and Technology Program (No.RCBS20210706092408010).

* M. Han, X. Sun and W. Zhan are with the School of Electronics and Communication Engineering, Shenzhen Campus of Sun Yat-sen University, Shenzhen, China (e-mail: hanmq@mail2.sysu.edu.cn; sunxinghua@mail.sysu.edu.cn; zhanw6@mail.sysu.edu.cn).

† X. Wang and X. Chen are with the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China (e-mail: wangxijun@mail.sysu.edu.cn; chenxiang@mail.sysu.edu.cn).

munication actions, as well as a nonlinear component for resource allocation. This MINLP problem introduces substantial computational complexity, making conventional optimization approaches impractical, particularly in scenarios with a large number of devices. Recent advances in Deep Reinforcement Learning (DRL) have demonstrated remarkable success in solving decision-making problems. By learning adaptive policies through environmental interactions, DRL frameworks can efficiently navigate the MINLP solution space, enabling better task offloading and resource allocation strategies in dynamic cloud-edge environments [24]–[36].

Within the management of massive industrial devices, there are several requirements in the ICPS, including the real-time, adaptability and performance requirements [1]–[7]. In cloud-edge computing networks, the real-time requirement aims at reducing the latency in deciding the task offloading and resource management decisions for all UEs. The adaptability requirement ensures that the cloud-edge computing network can provide reliable service for UEs with varying numbers and configurations. The performance requirement aims to optimize various objectives across devices, such as execution time for delay-sensitive tasks and energy cost for resource-limited UEs.

Considering these ICPS requirements, current DRL approaches face two problems in dynamic cloud-edge computing networks: 1) Joint optimization of task offloading and resource allocation across all UEs requires the DRL agent to operate in a hybrid high-dimensional state-action space, comprising discrete offloading decisions and continuous resource allocation. Such a complex state-action space poses challenges to conventional DRL schemes focusing on either discrete or continuous action space. 2) Conventional multi-layer perceptron (MLP) and Recurrent Neural Network (RNN)-based DRL models suffer from performance degradation when scaling to large networks due to their limited capacity to model complex state-action space. Moreover, these DRL models struggle to adapt to inexperienced scenarios with varying numbers of UEs due to the fixed input-output mapping, resulting in inferior adaptability [37]. 3) Conventional centralized DRL-based schemes depend on frequent global state synchronization between UEs, edge servers, and the cloud server. This leads to severe communication delay caused by the additional signaling overhead, particularly in large-scale cloud-edge computing networks.

To address the first and second challenges, we propose the Transformer-based Actor-Critic (TAC) model, which integrates the transformer architecture with an actor-critic framework. For joint task offloading and resource allocation, the TAC employs a hybrid actor that generates both the probability distribution of task offloading decisions and the continuous resource allocation strategy. These hybrid actions are then fed into the critic to compute Q-values of each state-action pair, which are then utilized to guide the learning of the hybrid actions. To handle the issue of varying numbers of agents, the transformer within the TAC efficiently captures relationships across N UEs while ensuring the output vector consistently includes resource management actions for all UEs, regardless of changes in N. This design allows the TAC model to seamlessly adapt to dynamic cloud-edge computing networks with fluctuating numbers of UEs.

To address the third problem, we propose to execute task offloading and resource management in a distributed manner, which can mitigate the severe communication delays caused by massive information exchange. Specifically, we propose a novel decentralized transfer learning (TL) method to transfer the centralized TAC model into a distributed one while maintaining the performance. During centralized training, the stacked-form joint state is adopted as global information, thereby enabling the centralized TAC model to manage task offloading and resource allocation for all UEs. After centralized training, we propose a decentralized TL approach for each device to independently make task offloading and resource management decisions based solely on local information, eliminating the need for information exchange between servers and UEs. Since both the joint state and the local state share the same size in the last dimension, the original TAC model can serve as the initialization point for the decentralized model. This design allows the model to retain the learned correlations across local states, significantly reducing the parameters required for retraining, while enhancing both convergence and performance. Moreover, unlike conventional TL methods that focus on mimicking the TAC model's actions, we leverage the converged critic in centralized TAC to guide the convergence of the distributed model.

A. Contribution and Main Results

To meet the real-time, adaptability, and performance requirements of ICPS, a distributed task offloading and resource management strategy is essential. In this paper, we propose a Distributed Transformer-based Actor-Critic (DTAC) algorithm designed to minimize both task execution time and energy consumption for resource-constrained UEs in a distributed manner. The main contributions are presented as follows:

- We propose the TAC architecture to address the joint task offloading and resource management problem in dynamic cloud-edge computing networks with a hybrid high-dimensional action space. To handle the hybrid action space, we introduce the hybrid actions of the actor, which serves as the input of the critic to obtain Q-values. These Q-values are utilized to guide the learning of both critic and actor, enabling the model to iteratively enhance the hybrid action decision. For the high-dimensional action space and dynamic scenarios, we employ the transformer to efficiently capture the intrinsic relationships between the UEs while adapting to inexperienced scenarios.
- We propose a novel decentralized TL approach to efficiently train the DTAC model by utilizing the converged centralized TAC model. Instead of minimizing the disparity between the policies of the DTAC and centralized TAC models, we leverage the learned critic model to provide the appropriate gradient information for training the DTAC model. This decentralized TL approach can significantly enhance both the convergence and performance, resulting in a performance level to that of the centralized TAC model.
- We evaluate the performance of DTAC in various scenarios, including its generalization capability in un-

trained environments. Compared with the state-of-the-art Heterogeneous-Agent Proximal Policy Optimization (HAPPO) in the MARL domain and conventional TL approaches, the DTAC approach demonstrates great generalization capability, and can achieve much better performance in cloud-edge computing networks with both various scales of coverage area and different numbers of UEs.

B. Related Work

The concept of task offloading and resource management was introduced to enable UEs to better utilize the communication and computation resources of edge servers, optimizing task execution time while saving energy [12], [13]. Due to the integer programming nature of this problem, DRL has emerged as a popular solution [24]–[36]. In [26], a deep deterministic policy gradient (DDPG)-based dynamic resource allocation strategy has been proposed to efficiently address the resource allocation problem with continuous state-action space. Similarly, by combining DDPG with hierarchical learning, the resource management problem in cloud-edge computing-based vehicular networks is addressed in [28]. In [29], the asynchronous advantage actor–critic algorithm is adopted to manage edge computing resources, enhancing cloud-edge computing performance while addressing security and privacy issues. For task offloading with integer actions, a double deep Q-network (DDQN)-based method is proposed to minimize energy consumption while considering the task delay constraint [24].

Other than using conventional MLP, transformer-based DRL schemes have recently been proposed for the task offloading problems to capture the correlations across UEs [32]–[36]. Unlike conventional MLP, the transformer can leverage the attention mechanism to better capture these correlations, thereby enhancing joint offloading and resource management decisions, particularly in large-scale and dynamic scenarios. For instance, in [32], a transformer-based Proximal Policy Optimization (PPO) approach was proposed for edge computing scenarios to jointly optimize offloading decisions, demonstrating superior performance and faster convergence compared to conventional DRL schemes. Furthermore, to facilitate model transfer across different environments, transformers have been integrated into both queuing delay prediction and actor networks due to their ability to handle varying input and output dimensions [34].

The aforementioned studies focus on optimizing offloading decisions in a centralized manner, resulting in significant signaling overhead and limiting their applicability in ICPS scenarios [10], [11]. Recently, multi-agent reinforcement learning (MARL) has achieved notable success in distributed decision-making problems. Through centralized training and decentralized execution (CTDE), agents can learn to cooperate in a distributed manner [38], [39]. In conventional CTDE approaches, the local states of all agents are concatenated into a global state to provide global information during centralized training, enabling agents to learn cooperative behavior using only local states during distributed execution. However, this approach cannot capture the intrinsic relationships between the states of the UEs and does not align well with the transformer architecture, leading to severe performance degradation [40]. Furthermore, in dynamic scenarios with varying numbers of agents, the concatenated global state must be padded with zero elements to create a fixed-length vector, causing a loss of positional information.

In this paper, we consider decentralized and centralized cloud-edge computing scenarios as related tasks, and propose a decentralized TL approach to transfer the centralized model into a distributed model. Recently, TL has shown remarkable proficiency in efficiently training new models for related tasks by leveraging prior knowledge, such as experiences and strategies, particularly in the DRL domain [41]–[44]. Utilizing this knowledge, agents can effectively learn to address current tasks that share similarities with past tasks. In the communication domain, TL leverages the correlation and similarity among tasks to significantly reduce training costs while enhancing convergence [42]. However, most TL approaches focus on student models that mimic the behavior of teacher models, such as actor models, while overlooking the critical role of the critic model. In the proposed decentralized TL, since both the joint state and the local state of the transformer share the same size in the last dimension, we propose to utilize the converged TAC actor model as the initialization point, thereby retaining the learned correlations across UEs to enhance convergence. Moreover, we further propose to employ the TAC critic model to guide its convergence towards a better direction, significantly improving the performance. After the decentralized TL, UEs can coordinate with each other in a distributed manner using only local information while maintaining a great performance level to that of the centralized model.

C. Outline

The remainder of this paper is organized as follows. In Sec. II, the system model and the problem of interest are described. In Sec. III, the decoupled optimization problem is formulated, and the proposed DTAC algorithm is illustrated. In Sec. IV, the performance of DTAC algorithm is evaluated. Finally, the paper is concluded in Sec. V.

II. System Model and Problem of Interest

A. System Model

As shown in Fig. 1, we consider a three-level cloud-edge computing network composed of a single cloud center, a set of edge servers $\mathcal{K}$ and a set of UEs $\mathcal{N}$. Each edge server is assumed to be deployed with a Base Station (BS) and covers multiple UEs. The set of UEs associated with edge server $k \in \mathcal{K}$ is denoted as $\mathcal{N}_k \subset \mathcal{N}$. For simplicity, we assume that UEs $n \in \mathcal{N}_k$ can only communicate with their associated edge server k. There are L channels in the network, in which each channel has the same bandwidth. The communication between UEs, edge servers and the cloud center is illustrated in Fig. 1. All UEs associated with the same edge server share the same channel resources, and each UE considers the transmission of other UEs using the same channel resource

This article has been accepted for publication in IEEE Journal on Selected Areas in Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2025.3574611

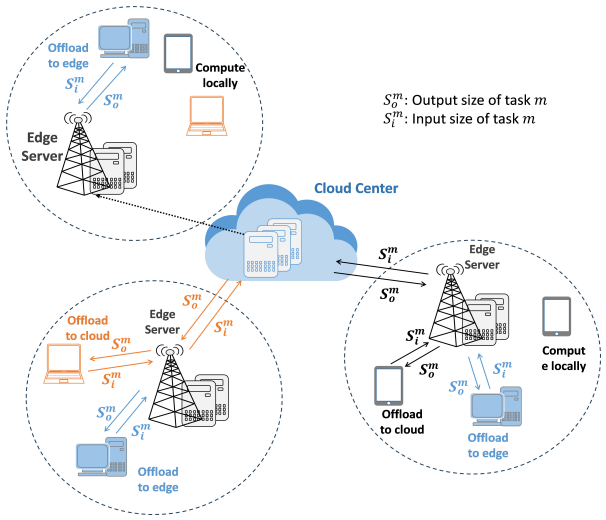


Fig. 1. The considered multi-user cloud-edge computing network with single cloud center, multiple edge servers and different types of UEs. The UEs associated with the same edge servers share the same channel resources, and the communications among edge servers and the cloud center share the same channel resources.

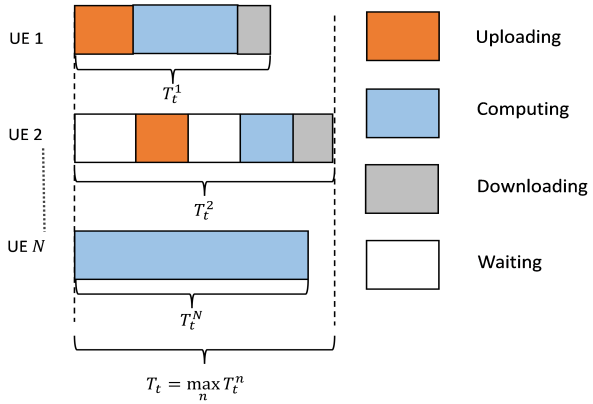


Fig. 2. Illustration of the workflow for the cloud-edge computing system. The duration of each decision time interval equals to the maximum computing time among UEs.

at the same time as interference. For simplicity, we consider that the communications of each edge server do not interfere with that of other edge servers due to non-overlap coverage of edge servers. Note that only edge servers can communicate with the cloud center, and tasks offloaded to the cloud center should be routed through the edge server as a transit.

In the considered cloud-edge computing network, we consider only UEs to have the task demand. The set of tasks is denoted as $\mathcal{M}$. Each computation task $m \in \mathcal{M}$ is represented by the vector (S_i^m, C^m, S_o^m), in which S_i^m and S_o^m denote the size of input data and the output computed data of task m, respectively, and C^m denotes the required CPU cycles per bit of input data [45]. In order to ensure successful joint resource allocation and decision-making for offloading and caching in the cloud-edge computing network, we divide time into rounds, as illustrated in Fig. 2. Each round begins with each UE assigned a single task randomly selected from the set of tasks $\mathcal{M}$. Note that different types of UE have their own independent task arrival distributions, resulting in varying expected numbers of different types of tasks arriving at different types of UEs. After receiving the assigned task, UEs make task offloading decisions, i.e., compute locally, upload to the edge server or further upload to cloud center. If a task is offloaded, the edge server or cloud center will process the task and send the result back to the UE n, requiring the UE to expend energy $E_{u,t}^n$ to upload the task. Alternatively, if a task is computed locally, the UE n can compute the task result locally without communication cost and delay, but with the expense of additional computation cost $E_{c,t}^n$. Each round ends only when all tasks have been computed, and the duration time of round t is denoted by T_t, which equals the maximum task execution time among all UEs.

Considering the real-time requirement for ICPS, we introduce the distributed cloud-edge computing network, in which each UE should independently execute the joint resource management, comprising task offloading position, local computation CPU frequency, communication channel selection, and corresponding transmission power. This distributed approach eliminates the need to collect the task and resource status of all UEs, thus avoiding signaling overhead. In particular, since we consider a distributed scenario where all actions are decided independently, each UE can only control its local computation frequency and transmission power between itself and its associated edge server. Meanwhile, the transmission power between edge servers and the cloud center, as well as the computation frequency of offloaded tasks, are controlled by the cloud center. Regarding the offloading decision, a task can either be computed locally, at the associated edge server, or at the cloud center. If the task is offloaded, the transmission channel between the UE and the edge server must also be determined by the UE to mitigate interference among UEs. Notably, as the task is offloaded to higher levels, computational resources become more abundant, but this comes at the cost of increased signaling overhead and communication delay.

In this paper, we aim at minimizing both the task execution time and the total energy cost of local UEs due to their limited energy resources. In the following, we will illustrate the details of the considered cloud-edge computing scenario and problem of interest.

B. *Resource Management*

For local communication resource, we denote the ratio of transmission power of each UE n to its maximum transmission power P_{UE} as $\mathbf{a_t} = \{a_t^n \mid n \in \mathcal{N}\}$, subject to $a_t^n \in (0, 1]$. Each UE n can select one of L orthogonal channels $l_t^n \in \{1, 2, \ldots, L\}$ for communication, aiming at mitigating the interference among UEs. For the computation resource, since the computational resource linearly relates to the CPU frequency, we consider the computation resource as the CPU frequency (in CPU rounds per second) in the following [20], [45]. The ratio of the local calculation frequency of each UE to the maximum CPU frequency F_{UE} is denoted as $\mathbf{b_t} = \{b_t^n \mid n \in \mathcal{N}\}$, subject to $b_t^n \in (0, 1]$. In particular, the calculation frequency and the transmission power of edge servers and the cloud

center are fixed as F_E, F_C and P_E, P_C, respectively. For the caching resources, we assume that only the edge servers have the capability to store the results of offloaded tasks. By caching the results of tasks, the edge servers can directly provide the cached task results to corresponding UEs, eliminating the need for redundant computations. The storage capacity of each edge server k is denoted as S^k (in bits). The edge server can cache the result of offloaded task m only when the current cache capacity is sufficient, in which the caching decisions of all edge servers are controlled by the cloud center.

Since the UEs have only limited computation resource, offloading the tasks to the associated edge server or the cloud center can significantly reduce the tasks computing time. The offloading decision of UE n at the round t is given by

$$x_t^n = \begin{cases} 2, \text{ if task is offloaded to the cloud center ,} \\ 1, \text{ if task is offloaded to nearest edge server ,} \\ 0, \text{ otherwise} \end{cases} . \quad (1)$$

In particular, when $x_t^n = 2$, since UEs $n \in \mathcal{N}_k$ can only communicate with its edge server k, the task will be transmitted from the edge server first, and then to the cloud center.

C. Computation Model

For the computation time, we first consider the caching status of edge servers. In the considered cloud-edge computing network, the edge server can cache the result of the offloaded task when its current cache capacity is sufficient and the task has been offloaded, i.e., $x_t^n \in \{1, 2\}$. The cache decision of each edge server k for each offloaded task m is decided independently using existing caching placement strategies, e.g., the least recently used and the least frequently used [46]. Then, we denote the cached results of tasks in edge server k as $c_t^k(m) \in \{0, 1\}$, where $c_t^k(m) = 0$ represents task m has not cached in edge server k at round t. Subsequently, the constraint of cache capacity is given by $S_k \geq \sum_m c_t^k(m) S_o^m$.

Then, we can derive the task computation time $\mathcal{T}_{1,t} = \left\{T_{1,t}^n \mid n \in \mathcal{N}\right\}$ according to the offloading decision x_t^n and caching status $c_t^k(m)$, i.e.,

$$T_{1,t}^n = \begin{cases} \frac{S_i^m C^m}{b_n^k F_{UE}}, & \text{if } x_t^n = 0, \\ (1 - c_t^k(m)) \frac{S_i^m C^m}{F_E}, & \text{if } x_t^n = 1, \\ \frac{S_i^m C^m}{F_C}, & \text{if } x_t^n = 2, \end{cases} , \quad (2)$$

and the corresponding energy cost $E_{c,t}$ for local computed tasks of UEs is given by [20]

$$E_{c,t}^n = \begin{cases} \kappa S_i^m C^m (b_t^n F_{UE})^2, & \text{if } x_t^n = 0, \\ 0, & \text{otherwise} , \end{cases} , \quad (3)$$

in which κ is the effective capacitance coefficient depending on the CPU chipset of UEs.

D. Communication Model

For the communication time, we first obtain the instantaneous data rates between UEs, edge servers and the cloud center, which are given by

$$R_t(n,k) = \begin{cases} B \log(1 + \frac{a_t^n P_{UE} |h(k,n)|^2}{\sigma^2 + P_{I,t}^n}), \text{from } n \text{ to } k, \\ B \log(1 + \frac{P_E |h(k,n)|^2}{\sigma^2}), \text{from } k \text{ to } n, \end{cases} , \quad (4)$$

$$R_t(c,k) = \begin{cases} B \log(1 + \frac{P_E |h(k,c)|^2}{\sigma^2}), \text{from } k \text{ to } c, \\ B \log(1 + \frac{P_C |h(k,c)|^2}{\sigma^2}), \text{from } c \text{ to } k, \end{cases}$$

where P_{UE}, P_E, and P_C denote the power of UE n, edge server k and the cloud center, respectively. B denotes the fixed bandwidth of each channel. $h(i,j)$ represents the channel gain between transmitter i and receiver j. For simplicity, we assume that each device has the same power for Gaussian noise, σ^2. Note that only transmissions from UEs to edge servers are interfered with by other UEs within the same coverage area, while transmissions initiated by edge servers and the cloud center are conducted through Time Division Multiple Access (TDMA) without interference. The power of the interference signal $P_{I,t}^n$ for UE n is given by:

$$P_{I,t}^n = \sum_{\substack{m \in \mathcal{N}_k \\ m \neq n}} \mathbb{1}\left(l_t^n = l_t^m\right) a_t^m P_{UE}, \quad (5)$$

in which $\mathcal{N}_k$ denotes the set of UEs associated with the same edge server as that of UE n, and $\mathbb{1}\left(l_t^n = l_t^m\right)$ represents that UE m offloads the task using the same channel as UE n. Subsequently, the energy cost $E_{u,t}$ for offloading the tasks from local UE n to the associated edge server is given by

$$E_{u,t}^n = \begin{cases} 0, & \text{if } x_t^n = 0, \\ a_t^n P_{UE} \frac{S_i^m}{R(n,k)}, & \text{if } x_t^n \in \{1, 2\}, \end{cases} , \quad (6)$$

in which $a_t^n P_{UE}$ is the transmission power and $\frac{S_i^m}{R(n,k)}$ represents the transmission time. Then, we denote the total energy cost E_t^n of each UE n at time t as

$$E_t^n = E_{u,t}^n + E_{c,t}^n. \quad (7)$$

Then, the task communication time between UEs and edge servers $\mathcal{T}_{21,t} = \{T_{21,t}^n | n \in \mathcal{N}\}$ can be derived as

$$T_{21,t}^n = \begin{cases} 0, & \text{if } x_t^n = 0, \\ \frac{S_i^m}{R(n,k)} + \frac{S_o^m}{R(k,n)}, & \text{if } x_t^n \in \{1, 2\}, \end{cases} , \quad (8)$$

and the communication time between edge servers and the cloud server $\mathcal{T}_{22,t} = \{T_{22,t}^n | n \in \mathcal{N}\}$ is given by

$$T_{22,t}^n = \begin{cases} 0, & \text{if } x_t^n \in \{0, 1\}, \\ \frac{S_i^m}{R(k,c)} + \frac{S_o^m}{R(c,k)}, & \text{if } x_t^n = 2. \end{cases} . \quad (9)$$

E. Queuing Model

Then, since the task offloading and computation procedure is sequentially executed over time, we denote the queuing matrix for computation on edge servers as $\mathcal{Q}_{11,t} = \{q_{11,t}(n_1, n_2) | n_1, n_2 \in \mathcal{N}\}$ and on the cloud center as $\mathcal{Q}_{12,t} = \{q_{12,t}(n_1, n_2) | n_1, n_2 \in \mathcal{N}\}$, where

$$q_{11,t}(n_1, n_2) = \begin{cases} 1, \text{ if } x_t^{n_1} = x_t^{n_2} = 1 \text{ and } n_1, n_2 \in \mathcal{N}_k, \\ 1, \text{ if } n_1 = n_2, \\ 0, \text{ otherwise} , \end{cases}$$

$$q_{12,t}(n_1, n_2) = \begin{cases} 1, \text{ if } x_t^{n_1} = 2 \text{ and } x_t^{n_2} = 2, \\ 1, \text{ if } n_1 = n_2, \\ 0, \text{ otherwise} , \end{cases} . \quad (10)$$

Then, we denote the queuing matrix for communication as $\mathcal{Q}_{2,t} = \{q_{2,t}(n_1, n_2) | n_1, n_2 \in \mathcal{N}\}$, which is given by

$$q_{2,t}(n_1, n_2) = \begin{cases} 1, \text{ if } x_t^{n_1} = x_t^{n_2} = 1 \text{ and } n_1, n_2 \in \mathcal{N}_k, \\ 1, \text{ if } x_t^{n_1} = 2 \text{ and } x_t^{n_2} = 2, \\ 0, \text{ otherwise }, \end{cases} . \quad (11)$$

In particular, $\mathcal{Q}_{11,t}$, $\mathcal{Q}_{12,t}$ and $\mathcal{Q}_{2,t}$ represent that UE n_1 and n_2 needs to wait for each other to complete task computing and communication through a sequential manner. Finally, we can obtain the duration time T_t of each round t as

$$\begin{aligned} T_t = &\|\mathcal{T}_{1,t}(0) + \mathcal{Q}_{11,t} \cdot \mathcal{T}_{1,t}(1) \\ &+ \mathcal{Q}_{12,t} \cdot \mathcal{T}_{1,t}(2) + \mathcal{Q}_{2,t} \cdot \mathcal{T}_{21,t}(1) + \mathcal{T}_{22,t}(2)\|_\infty , \end{aligned} \quad (12)$$

where $\mathcal{T}_t(x) = \{T_t^n \text{ if } x_t^n = x \text{ else } 0 | n \in \mathcal{N}\}$ represents a part of matrix $\mathcal{T}_t$ comprising by the elements of specific offloading decision x.

F. Objective Function

In cloud-edge computing networks, a swift task execution is crucial to meet the demand of delay-sensitive applications. In the meanwhile, since UEs with limited resources cannot handle excessive energy demands for every computing task, maintaining low energy cost is equally important. In this paper, we aim to optimize both the task execution performance of the cloud-edge computing network and energy consumption, while adhering to the constraints of cache, computation, and communication resources. By denoting $\mathbf{x_t} = \{x_t^1, \ldots, x_t^N\}$, $\mathbf{l_t} = \{l_t^1, \ldots, l_t^N\}$, $\mathbf{a_t} = \{a_t^1, \ldots, a_t^N\}$ and $\mathbf{b_t} = \{b_t^1, \ldots, b_t^N\}$, and letting $\mathbf{x} = \{\mathbf{x_t}\}_t^T$, $\mathbf{l} = \{\mathbf{l_t}\}_t^T$, $\mathbf{a} = \{\mathbf{a_t}\}_t^T$ and $\mathbf{b} = \{\mathbf{b_t}\}_t^T$, the optimization problem can be formulated as

$$\begin{aligned} &\underset{\mathbf{x},\mathbf{l},\mathbf{a},\mathbf{b}}{\text{minimize}} \quad \lim_{T\to\infty} \frac{1}{T} \cdot \sum_{t=1}^{T} (\omega_1 T_t + \omega_2 \sum_{n=1}^{N} E_t^n) \\ &\text{subject to:} \\ &a_t^n \in (0,1], b_t^n \in (0,1], x_t^n \in \{0,1,2\}, l_t^n \in \{1,2,\ldots,L\} \end{aligned} , \quad (13)$$

in which ω_1 and ω_2 denote the weight of the average task execution time T_t and the total energy cost of all UEs $\sum_{n=1}^{N} E_t^n$, respectively.

III. DTAC Algorithm

In this paper, we propose the DTAC algorithm, in which a centralized TAC is first trained to coordinate the task offloading and resource management decisions of all UEs, and a decentralized TL is then proposed to efficiently transfer the converged centralized transformer-based model into a distributed one. In the following, we will sequentially demonstrate the centralized training, the network architecture and loss function of the TAC model, the proposed decentralized TL and the DTAC model.

A. DRL Model Formulation

In this section, we illustrate the Partially Observable Markov Decision Process (POMDP) of the considered cloud-edge computing network within both centralized training and decentralized TL. In the POMDP, each agent first observes the environmental state, makes the offloading and resource management decision, and then receives the reward.

A.1 State

As aforementioned, the transformer has the feature that the number of outputs keeps the same as inputs utilizing the encoder-decoder architecture. By introducing the transformer to address the high-dimensional state and action space, we define the state of each UE and aggregate them to form the agent's state as input of the TAC model. Since the communication time of the task computed locally is always given by $T_{2,t}^n(0) = 0$, we propose the state of each UE n as

$$\begin{aligned} s_t^n = \{&P_{UE}|h(k,n)|^2, T_{1,t}^{*,n}(0), T_{1,t}^{*,n}(1), \\ &T_{21,t}^{*,n}(1), T_{1,t}^{*,n}(2), T_{22,t}^{*,n}(2)\} \in \mathbb{R}^6, \end{aligned} \quad (14)$$

in which $T_{1,t}^{*,n}(x), T_{21,t}^{*,n}(x), T_{22,t}^{*,n}(x)$ denotes the value of $T_{1,t}^n$, $T_{21,t}^n$ and $T_{22,t}^n$ of (2), (8) and (9) given by $x_t^n = x$, $a_t^n = b_t^n = 1$, respectively. Specifically, $T_{1,t}^{*,n}(0), T_{1,t}^{*,n}(1), T_{21,t}^{*,n}(1)$, $T_{1,t}^{*,n}(2), T_{22,t}^{*,n}(2)$ represent the lower bound of computation and communication time for computing locally, offloading to edge servers and cloud center without the interference and queuing of other UEs, respectively. In particular, each local state s_t^n takes both the task status and resource information for local UE, edge servers and cloud center into consideration, thereby providing sufficient information for making independent task offloading decisions. $P_{UE} \cdot |h(k,n)|^2$ represents the power of the received signal at the associated edge server k.

Accordingly, the joint state for centralized TAC agent $\mathbf{s_t}$ is given by

$$\mathbf{s_t} = \{s_t^n\}_{n=1}^N \in \mathbb{R}^{N\times 6}. \quad (15)$$

A.2 Action

In the considered cloud-edge computing network, each UE should decide its own offloading decision $x_t^n \in \{0,1,2\}$, transmission channel $l_t^n \in \{1,2,\ldots,L\}$ and corresponding transmission power $a_t^n \in (0,1]$ and calculation frequency $b_t^n \in (0,1]$. During centralized training, the joint action $\{\mathbf{x_t}, \mathbf{l_t}, \mathbf{a_t}, \mathbf{b_t}\} \in \mathbb{R}^{N\times(L+5)}$ of all UEs is decided centrally using the global state $\mathbf{s_t}$. In particular, we use the probability $p_{x,t}, p_{l,t}$ to determine the final offloading decision and channel selection $\mathbf{x_t}$, $\mathbf{l_t}$. Subsequently, the overall action of the TAC model is denoted as

$$A_t = \{p_{x,t}, p_{l,t}, \mathbf{a_t}, \mathbf{b_t}\} \quad (16)$$

A.3 Reward

As our goal is to minimize both the execution time T_t and energy cost E_t in each round t under the joint caching, communication, and computation resources constraints, we propose a vector-form reward for each UE to enhance the sample efficiency. Towards the objective function, we use the negative weighted value of task execution time and energy cost as the reward, which is given by:

$$r_t = \{-\omega_1 \cdot \{T_t^n\}_{n\in\mathcal{N}}, \ -\omega_2 \cdot \{E_t^n\}_{n\in\mathcal{N}}\} \in \mathbb{R}^{2N}. \quad (17)$$

Here, ω_1 and ω_2 denote the weight of execution time T_t and energy cost E_t^n, respectively. Towards this multi-objective problem, a conventional approach is to sum up the rewards of different objectives in a scalar. However, such an approach will flatten the model gradient and increase the risk for convergence to local optimums [47]. Considering the transformer

This article has been accepted for publication in IEEE Journal on Selected Areas in Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2025.3574611

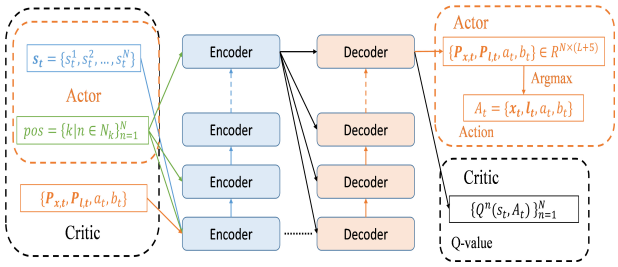


Fig. 3. Architecture of the TAC model. Actor takes the joint state $\mathbf{s}_t$ as input to obtain action A_t. Critic takes both joint state $\mathbf{s}_t$ and action A_t as input to obtain Q-values $Q(\mathbf{s}_t, A_t)$. The hidden state is first passed through each encoder, followed by each decoder in a sequential manner. In particular, *pos* serves as the position embedding.

architecture of DTAC, we propose a vector-form reward. For the local reward of each UE, we divide it into two parts $r_t^n = \{-\omega_1 T_t^n, -\omega_2 E_t^n\}$. Subsequently, we stack the task execution time and energy cost for all UEs into a vector-form reward r_t instead of using the scalar-form objective function as a reward, i.e., $\omega_1 T_t + \omega_2 \sum_{n=1}^{N} E_t^n$. This mechanism can enable the critic to evaluate the individual Q-value of each UE, thereby avoiding the flattened gradient issue and leading to improved sample efficiency and convergence [48].

Through this reward design, the convergence performance of TAC can be significantly enhanced, and the optimization problem in (13) can be addressed when the proposed TAC model converges.

B. Transformer Architecture

As aforementioned, we introduce the transformer architecture in the proposed TAC algorithm instead of the widely used MLP in other DRL approaches, which has two-fold reasons. From the perspective of the input-output dimension, given the input state dimension $\mathbf{s_t} \in \mathbb{R}^{N\times 6}$, the transformer ensures the action dimension $A_t \in \mathbb{R}^{N\times(L+5)}$ regardless of the number of UEs N. In contrast, conventional MLP requires flattening the joint state into a one-dimensional vector of length $6 \cdot N$ to capture the correlation among local states s_t^n [49]. Due to the mobility of UEs, the number of UEs in the cloud-edge computing system is highly dynamic, leading to a time-varying value for N. Since the size of the input vectors for MLP depends on N, it cannot be applied in dynamic scenarios with varying numbers of UEs. Furthermore, MLP faces challenges in calculating the joint action for all UEs since it is difficult to represent this in a one-dimensional vector format. For MLP, directly transitioning its one-dimension output vector $A_t \in \mathbb{R}^{N(L+5)}$ into a two-dimension action $A_t \in \mathbb{R}^{N\times(L+5)}$ may encounter severe performance degradation since it requires reshaping the output in a way that may not preserve the spatial and temporal relationships inherent in the input. The flattened input and output can result in the loss of local dependencies between UEs, thereby hindering the performance. From the perspective of architecture, the MLP captures the correlation of each element of input, which includes a significant amount of irrelevant information and neglects the topology of the cloud-edge computing system. In contrast, the transformer efficiently captures the inherent correlations across UEs through adaptive attention mechanisms. Additionally, the transformer can incorporate the system topology by using position embedding, further enhancing its ability to model the relationships between the components.

The TAC architecture is illustrated in Fig. 3. In the actor network, the joint state $\mathbf{s_t}$ is passed through each encoder in a sequential manner, while position embedding *pos* serves as position embedding. In particular, since the UEs associated with the same edge server exhibit coupling computation and communication time when offloading $x_t^n \in \{1, 2\}$, we adopt the id of associated edge server k of each UE $n \in \mathcal{N}_k$ as the position embedding to capture the intrinsic relationships among UEs. Then, the output of encoders are passed to the decoders, and then output the probability $p_{x,t} = \{p_{x,t}^n\}_{n\in\mathcal{N}} \in \mathbb{R}^{N\times 3}$ of each offloading decision $x_t^n \in \{0, 1, 2\}$, the probability $p_{l,t} = \{p_{l,t}^n\}_{n\in\mathcal{N}} \in \mathbb{R}^{N\times L}$ of each offloading decision $l_t^n \in \{1, 2, \ldots, L\}$, and directly output the value of $\mathbf{a_t} = \{a_t^1, \ldots, a_t^N\}$ and $\mathbf{b_t} = \{b_t^1, \ldots, b_t^N\}$ for all UEs.

In particular, other than sampling actions from the probability $p_{x,t}$, we select the action with maximum probability as the offloading decision in a deterministic manner, i.e., $\mathbf{x_t} = \underset{x_t^n\in\{0,1,2\}}{\arg\max}\, p_{x,t}$ and $\mathbf{l_t} = \underset{l_t^n\in\{1,2,\ldots,L\}}{\arg\max}\, p_{l,t}$, instead of using the probability to sample actions. The reason is two-fold. First, the output of the actor $p_{x,t}, p_{l,t}$ also serves as the input of critic network, which is used to evaluate the Q-value of each state-action pair $Q(\mathbf{s_t}, \mathbf{A_t})$ and guides the convergence direction of action during training. Moreover, we further propose a vector-form Q-value as similar to the vector-form reward r_t. However, in the high-dimensional action space, randomly selecting action $\mathbf{x_t} \in \mathbb{R}^{N\times 3}$, $\mathbf{l_t} \in \mathbb{R}^{N\times L}$ according to the probability distribution $p_{x,t}, p_{l,t}$ will hinder the evaluation on the Q-value space. Since there are $(L+3)^N$ types of offloading actions that can be obtained from certain probabilities, the random sample approach would significantly increase the complexity of training data collection and lead to a much lower convergence rate. In contrast, the proposed deterministic approach can address this issue of high-dimensional action space by providing a direct mapping between $p_{x,t}$ and $\mathbf{x_t}, p_{l,t}$ and $\mathbf{l_t}$.

Second, since the critic network takes the action probability $p_{x,t}, p_{l,t}$ as input and passes the Q -value to guide the convergence direction of the actor, randomly sampling action may counterintuitively degrade the exploration ability of actor and leads to convergence in unexpected local optimum. In the random approach, the output Q -value $Q(\mathbf{s_t}, \mathbf{A_t})$ is the expected value of reward with respect to the action probability, and critic will guide the actor converge to those experienced actions with largest reward in each state, resulting in the probabilities converging to a one-hot vector. Such an approach has poor exploration ability even using both ϵ-greedy and Gaussian noise. In the proposed deterministic approach, the TAC only needs to ensure that the probability of the action with the maximum Q-value is higher than the others, rather than maximizing this probability to approach one. Utilizing the Gaussian noise, this approach can significantly enhance the exploration capability and convergence.

For the critic network, as presented in the black part of the Fig. 3 each encoder takes not only the state $\mathbf{s_t}$ and position embedding pos, but also the action probability $p_{x,t}$, $p_{l,t}$ and allocation ratio a_t, b_t as input. In particular, we take p_t as input of critic instead of $\mathbf{x_t}$ since the gradient cannot be directly passed through the non-differentiable $\arg\max$ operation. Then, the critic outputs the Q-value $\{Q^n(\mathbf{s_t}, A_t)\}_{n\in\mathcal{N}}$ for all UEs. In particular, the Q-values $\{Q^n(\mathbf{s_t}, A_t)\}_{n\in\mathcal{N}}$ aim to evaluate the task execution time and energy cost of each UE n, in which all experienced data can be utilized to accelerate the convergence. Finally, since we aim at minimizing the maximum task execution time at each round as presented in (12), the maximum Q-value among UEs $\max_n\{Q^n(\mathbf{s_t}, A_t)\}_{n\in\mathcal{N}}$ acts as the loss function of actor network.

C. Algorithm Overview

Algorithm 1 Distributed Transformer Actor-Critic algorithm

Require: Initial variance of noise v, decay weight of noise ϵ, rounds per episode L, actor network, critic network.
Collecting data:
for Round $t \le L$ **do**
  Obtain action probability p_t^n according to state $\mathbf{s_t}$ for each UE n
  Select the actions with maximum probability using Gaussian noise, $a_t^n = \arg\max p_t^n + \mathcal{N}(0, v)$ for each UE n
  Receive reward vector r_t for all UEs, and store the experience $\{\mathbf{s_t}, \mathbf{p_t}, r_t\}$ into memory
Centralized Training:
while Centralized model has not converged **do**
  for U times updating per episode **do**
    Randomly sample a batch of experiences $\{\mathbf{s_t}, \mathbf{A_t}, r_t\}$
    Output the Q-values of all UEs through critic, and update critic parameters using critic loss in (18)
    Take the maximum Q-value as actor loss (19) to update actor parameters
  Update the variance of Gaussian noise every episode $v = v \cdot \epsilon$
Decentralized Transfer Learning:
Create a copy of the centralized TAC model π^* as the DTAC model π'
Change the attention procedure from $\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ to $\left(\frac{I}{\sqrt{d_k}}\right)V$
while Actor loss in (19) has not converged **do**
  Randomly sample a batch of experiences $\{\mathbf{s_t}, \mathbf{A_t}, r_t\}$
  Obtain the output A_t of DTAC model π'
  Minimize the actor loss in (19)
Distributed Execution:
for Each UE $n \le N$ **do**
  Obtain action A_t^n using only the local state s_t^n and the converged DTAC model π' to manage the joint resources independently

The training procedure of the proposed TAC algorithm is illustrated in Fig. 4, and the detailed outline presented in Algorithm 1. In the following, we illustrate the training procedure from the perspective of the centralized training for TAC model and the decentralized TL for DTAC model, respectively.

C.1 Centralized Training for TAC

During centralized training, the centralized TAC agent first gathers the states of all UEs as $\mathbf{s_t} = \{s_t^1, \ldots, s_t^N\}$, and then determines the offloading decision $\mathbf{x_t}$, transmission channels $\mathbf{l_t}$, and resource allocation $\mathbf{a_t}$, $\mathbf{b_t}$. In particular, Gaussian noise is introduced in the action probabilities $p_{x,t}$ and $p_{l,t}$ of all UEs to enhance the exploration ability during training. Subsequently, the TAC agent receives a reward r_t from the environment and stores the experiences $\mathbf{s_t}, x_t, \mathbf{l_t}, a_t, b_t$, and r_t in memory for subsequent training iterations. After collecting sufficient experiences, the critic network initially processes the vector-form reward $\mathbf{r}_t$ of all UEs to evaluate their Q-value $Q(\mathbf{s_t}, A_t)$ and provides estimated gradients for the actor, resulting in faster convergence rates compared to using only scalar-form rewards. The maximum value of $Q(\mathbf{s_t}, A_t)$ is then utilized to update the model parameters of the actor network through gradient descent. Given that the Q-values are intended to accurately assess the task execution time and energy cost of all UEs, we employ the Mean Squared Error (MSE) loss function as the critic's loss., i.e.,

$$L_{\text{critic}} = \sum_{n\in\mathcal{N}} \left(Q(\mathbf{s_t}, A_t) - r_t\right)^2 \tag{18}$$

in which $Q(\mathbf{s_t}, A_t)$ denotes the Q-value representing the expected reward of both task execution time and energy cost, given the action $p_{x,t}$, $p_{l,t}$, a_t, b_t in the specific state $\mathbf{s_t}$ of all UEs. In particular, we introduce p_t instead of x_t to address the problem that the $\arg\max$ function is non-differentiable, which enables the gradient of critic can be passed to the actor for gradient descent. With the convergence of the critic network, it can evaluate the Q-value $Q(\mathbf{s_t}, A_t)$ more accurately, and provide a better approximated gradient of the non-differentiable objective function T_t and E_t^n.

Subsequently, the actor can utilize the gradient $\partial \max_n Q(\mathbf{s_t}, A_t)/\partial x$ to efficiently optimize the offloading policy. The loss function of actor is given by,

$$\begin{aligned} L_{actor} = \nabla_\pi \pi(A_t \mid \mathbf{s_t}) \nabla_{A_t}[\omega_1 \max_n Q(\mathbf{s_t}, A_t)_{1,:} |_{A_t=\pi(A_t|\mathbf{s_t})} + \\ \omega_2 \sum_n Q(\mathbf{s_t}, A_t)_{2,:} |_{A_t=\pi(A_t|\mathbf{s_t})}] - \sum_{n\in\mathcal{N}} \omega_s S[\pi(A_t \mid \mathbf{s_t})], \end{aligned} \tag{19}$$

where $Q(\mathbf{s_t}, A_t)_{1,:}$ and $Q(\mathbf{s_t}, A_t)_{2,:}$ represent the evaluated task execution time and energy cost given specific state $\mathbf{s_t}$ and action A_t, respectively. In particular, $\max_n Q(\mathbf{s_t}, A_t)_{1,:}$ represents the evaluated value of T_t since we take the maximum task execution time among agents as the duration time of round t. $\sum_n Q(\mathbf{s_t}, A_t)_{2,:}$ represents the total energy cost of all UEs. $\pi(A_t, \mathbf{s_t})$ denotes the actor model. $\omega_s S[\pi(A_t, \mathbf{s_t})]$ represents the entropy-loss of the current policy given state $\mathbf{s_t}$ as similar to that in the conventional Soft Actor-Critic (SAC) algorithm, which is introduced to better explore the state-arm space during training.

C.2 Decentralized TL for DTAC

After the centralized training of TAC model, we propose the decentralized transfer learning approach to transfer the centralized TAC model to the DTAC model, which can decide action independently using only local state of each UE. In the centralized TAC model, the multi-head attention mechanism aggregate the global information among UEs

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{20}$$

in which Q, K, V are the query, key and value of the joint input state $\mathbf{s_t}$ of all UEs, respectively. In particular, the value of $\left(\frac{QK^T}{\sqrt{d_k}}\right)_{[i,j]}$ represents the attention weight of UE i paid on the UE j. Through such attention mechanism, each UE can make better resource management decision using not only local state but also information of other UEs.

This article has been accepted for publication in IEEE Journal on Selected Areas in Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2025.3574611

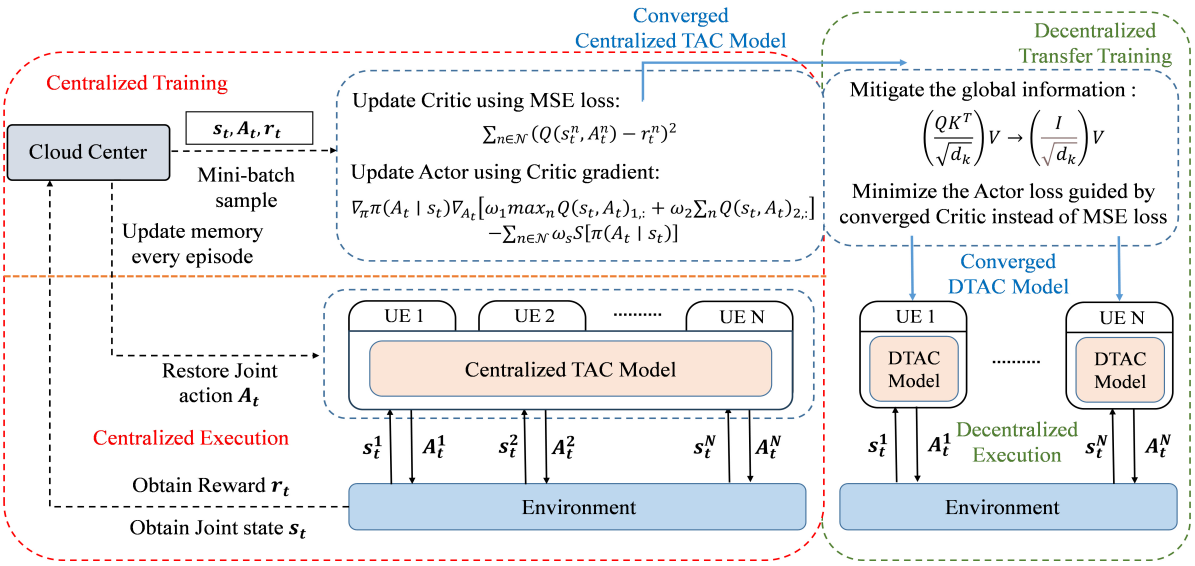


Fig. 4. Illustration of the TAC procedure. The critic network uses collected data to train and enhance the evaluation ability on Q-values, and the actor network takes the maximum value of Q-values as a loss function to perform gradient descent. First, the TAC model is trained to optimize the resources management of all UEs in a centralized manner. Then, the converged TAC model is further trained into a DTAC model by reducing the weight of context information of other UEs, and then is applied for resource management in a decentralized manner.

In the proposed DTAC, we aim to enable the independent decision ability for all DTAC agents. Therefore, each DTAC agent n should only pay attention to its own state s_t^n instead of taking the joint state $\mathbf{s_t}$ into consideration, and make local action A_t^n independently without the requirement of global information. According to (20), when deciding the action of UE i, the centralized TAC model gathers the information of other UEs using the weight $\left(\frac{QK^T}{\sqrt{d_k}}\right)_{[i,j]}$. In the proposed DTAC model, we transfer the procedure of attention block that only utilizes the value V and ignores the query Q and key K, enabling the ability for independent decision using only local state for all UEs,

$$\text{Attn}(Q, K, V)' = I \cdot V. \tag{21}$$

In (21), each UE only pays attention to its individual state s_t^n while ignoring the information of others, enabling each UE to decide task offloading and resource management independently. In particular, we focus on transferring the actor model of DTAC during decentralized TL while keeping the centralized critic model unchanged. Instead of the MSE loss between the policies of DTAC and TAC model widely applied in conventional TL approaches, we propose to train the DTAC model using the converged critic network in the aforementioned TAC model, which is given by

$$\begin{aligned} L'_{\text{actor}} = \nabla_\pi \pi\left(A'_t, \mathbf{s_t}\right) \nabla'_{A_t} \Bigg[\omega_1 \max_n Q\left(\mathbf{s_t}, A'_t\right)_{1,:} \Big|_{A_t^n = \pi(A_t^n \mid s_t^n)} + \\ \omega_2 \sum_n Q\left(\mathbf{s_t}, A'_t\right)_{2,:} \Big|_{A_t^n = \pi(A_t^n \mid s_t^n)} \Bigg] - \sum_{n\in\mathcal{N}} \omega_s S\left[\pi\left(A'_t, \mathbf{s_t}\right)\right] \end{aligned} \tag{22}$$

Here, $A'_t = \{A_t^1, \ldots, A_t^N\}$ represents the stacked distributed actions of each UE, in which each local action A_t^n is determined based only on local information $\pi\left(A_t^n \mid s_t^n\right)$. Upon the convergence of the DTAC model, each UE n can independently make resource management decision A_t^n using only its local state s_t^n.

The reason for introducing the converged critic in (22) is two-fold. First, the centralized actor of TAC also aims to maximize the expected Q-value estimated by the critic. Therefore, by leveraging the accurate gradient information provided by the converged critic model, the DTAC model can effectively optimize its parameters towards a better direction compared to the MSE loss. Second, the MSE loss only encourages decentralized models to replicate the behavior of the centralized TAC without providing additional inherent information, i.e., the underlying system dynamics or the interaction between UEs. Therefore, using MSE loss makes the DTAC model more likely to converge to a local optimum, e.g., $A_t^n = \frac{1}{n}\sum_n A_t^{*,n}$. Moreover, using MSE loss for behavior cloning further encounters severe issues when data is limited. Since the behavior cloning mainly relies on the experience during decentralized TL, it performs well only within the state distribution of this experience. However, during practical testing, any deviations can cause compounding errors, leading to poor performance in real environments [50].

IV. Simulation Results and Discussions

In this section, we evaluate the proposed TAC algorithm by comparing it with other methods. In the following, we first introduce the simulation setup, and then present the detailed performance evaluations under different scenarios.

A. Simulation Setting

In simulations, the number of UEs associated with each edge server follows the uniform distribution, in which their positions follow a two-dimensional uniform distribution taking

the position of the edge server as center. Edge servers are uniformly distributed on a fixed-radius circumference around the cloud center. The parameters of cloud-edge computing network and hyper-parameters of DTAC are presented in Table I and Table II, respectively.

TABLE I
CLOUD-EDGE COMPUTING NETWORK PARAMETERS [1]

Parameters	Value
Radius of edge server	$\{200, 300, 600\}$ meters
Radius of Cloud Center	$\{300, 400, 800\}$ meters
Number of edge servers	6 or 8
Number of UEs within an edge server	$\{3, 4, 5\}$
Number of types of tasks	$M = 200$
Number of types of UEs	6
Total Bandwidth	80 MHz
Total Computation Resources	86.4 or 115.2 GHz
Transmission Power P_C, P_E, P_{UE}	$40, 30, 23.9$ dBm
Distribution of input data size	$[5, 20]$ $Mbits$
Distribution of output data size	$[2, 40]$ $Mbits$
Distribution of CPU rounds	$[600, 1200]$

TABLE II
HYPER-PARAMETERS OF DTAC

Parameters	Value	Parameters	Value
Learning rate	$2 \cdot 10^{-5}$	Number of rounds	600
Optimizer	AdamW	Entropy coefficient ω_s	0.01

To present a comprehensive comparison, the following approaches are used to evaluate the performance of the proposed DTAC algorithm.

- *HAPPO:* We introduce the HAPPO as the baseline of conventional MARL approaches, in which the popular RNN is deployed. For the POMDP setting of HAPPO, the reward of each UE n is given by

$$r_t^n = \omega_1 T_t + \omega_2 \sum_{n=1}^{N} E_t^n, \tag{23}$$

  which corresponds to the objective function presented in (13) and shares the same value among UEs. In particular, designing a UE-specific reward, e.g., $\omega_1 T_t + \omega_2 E_t^n$, may lead to performance deterioration in our simulations. In local state of HAPPO, we incorporate the id of each UE n and its associated edge server k for identification,

$$\begin{aligned} s_t^n = \{n, k, P_{UE}|h(k,n)|^2, T_{1,t}^{*,n}(0), T_{1,t}^{*,n}(1), \\ T_{21,t}^{*,n}(1), T_{1,t}^{*,n}(2), T_{22,t}^{*,n}(2)\} \in \mathbb{R}^8. \end{aligned} \tag{24}$$

  Finally, we take the concatenation of local states as the global state during centralized training

$$s_{t,g} = \{s_t^1, s_t^2 \ldots, s_t^N, 0, 0, \ldots 0\} \in \mathbb{R}^{8 \cdot N_{max}}, \tag{25}$$

  in which we take zero vector as the padding of $s_{t,g}$ to ensure the application of HAPPO in scenarios with the number of UEs ranging from N_{min} to N_{max}.
- *TL:* We introduce the TL as the baseline of conventional TL approaches. In the TL, we apply the same local state as that of HAPPO in (24), and take the same actor model in the HAPPO as the agents' model in TL. During centralized training, we apply the converged centralized TAC model as the teacher model to generate the labeled data, and take the TL model as the student model to learn the optimal actions of the teacher centralized TAC model using MSE loss

$$L_{\text{TL}} = \frac{1}{|B|} \sum_{t \in B} \left(\{A_t^n\}_{n \in \mathcal{N}} - A_t^* \right)^2 \tag{26}$$

  in which B denotes the sampled batch. $\{A_t^n\}_{n \in \mathcal{N}}$ is the joint action of the student TL model using local state of each UE, and A_t^* is the action of centralized TAC model. In particular, TL has the same labeled data of that during training the DTAC model.
- *TL transformer:* We also introduce a TL approach that adopts the same model as that of the TAC approach for a better comparison. The only difference between this approach with the proposed DTAC approach is that we optimize the distributed actions of the DTAC models using the MSE loss illustrated in (26).
- *Random:* We also adopt the random approach to serve as the worst performance among approaches. In the Random approach, both the offloading actions x_t^n and the channel selections l_t^n for all UEs are randomly selected from $x_t^n \in \{0, 1, 2\}$ and $l_t^n \in \{1, 2, \ldots, L\}$, respectively. The ratio of communication power $a_t^n \in (0, 1]$ and computation frequency $b_t^n \in (0, 1]$ are uniformly distributed.

B. Performance Metrics

Considering performance requirement for resources-limited UEs in ICPS, the following metrics are used to evaluate the proposed DTAC.

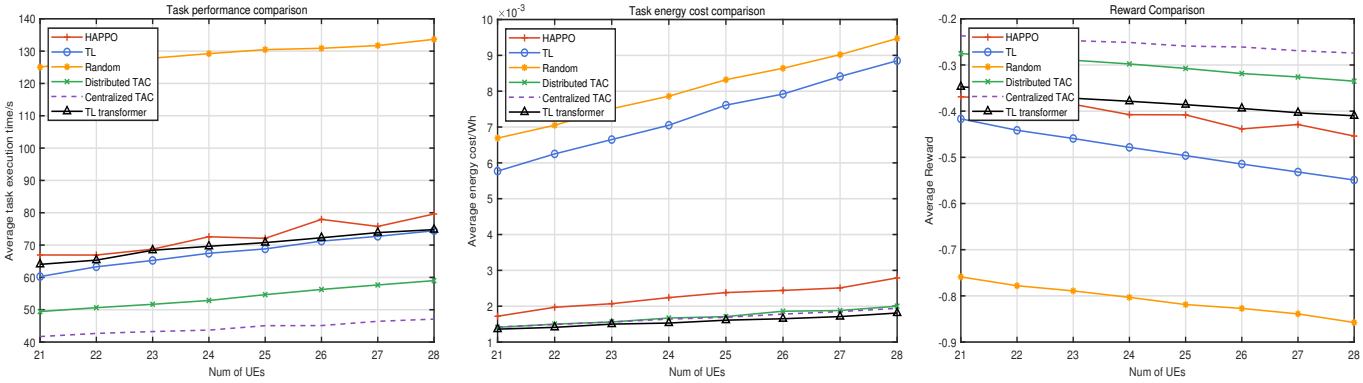
- *Average task execution time:* We demonstrate the average task execution time among rounds in each episode to evaluate the task performance of the proposed DTAC approach
- *Average energy cost:* We demonstrate the average energy cost among UEs in each episode to evaluate the energy performance of the proposed DTAC approach.
- *Weighted reward*: We demonstrate the cloud-edge computing network performance including both energy cost and task execution time to evaluate the overall performance of the proposed DTAC approach. The weighted reward is the negative value of objective function, which is given by

$$-\frac{1}{T} \cdot \sum_{t=1}^{T} \left[\omega_1 T_t + \omega_2 \sum_{n=1}^{N} E_t^n \right] \tag{27}$$

C. Performance Comparison

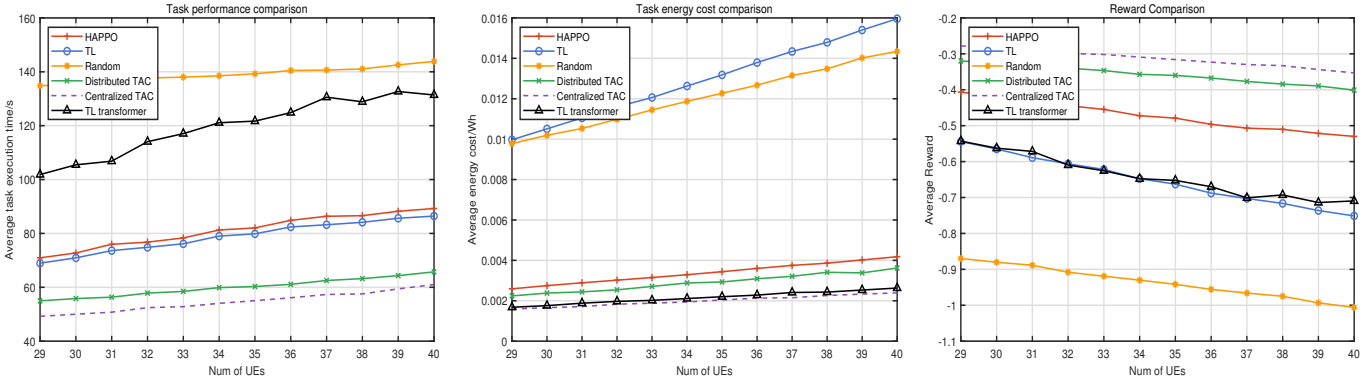
In this section, we first evaluate the proposed DTAC in scenarios comprising one cloud center, 5 edge servers, and a variable number of UEs associated with each edge server, ranging from 4 to 6. The performance comparison on average task execution time, the average energy cost per task

This article has been accepted for publication in IEEE Journal on Selected Areas in Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2025.3574611



(a) Average task execution time. (b) Average energy cost. (c) Average reward.

Fig. 5. Performance comparison between the proposed DTAC and other approaches in scenario with 5 edge servers, in which the coverage radius of edge server is $200m$ and that of cloud center is $300m$.. (a) illustrates the average task execution time comparison; (b) illustrates the average energy cost per task; (c) illustrates the weighted reward corresponding to the objective function $-\frac{1}{T}\cdot\sum_{t=1}^{T}\omega_1 T_t+\omega_2\sum_{n=1}^{N}E_t^n$.



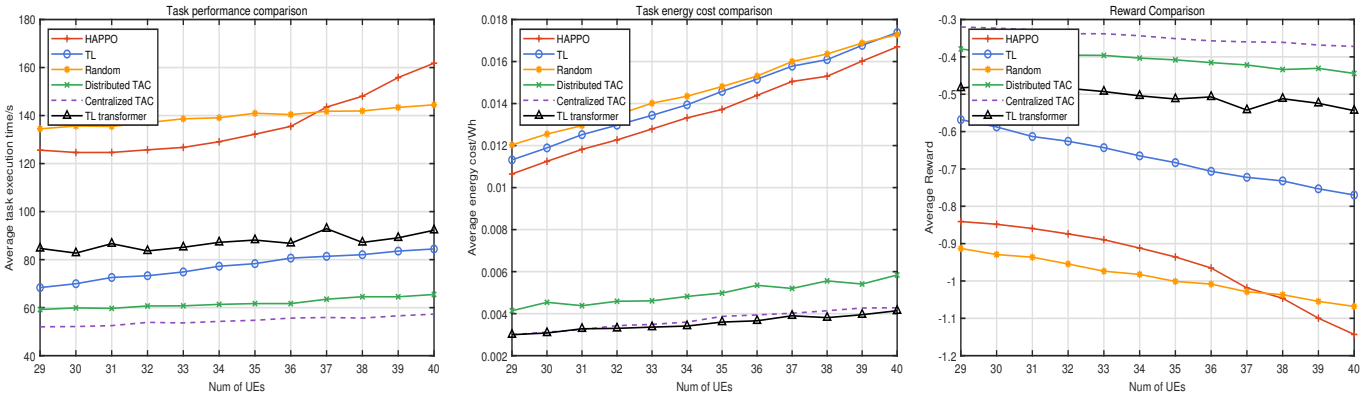
(a) Average task execution time. (b) Average energy cost. (c) Average reward.

Fig. 6. Performance comparison between the proposed DTAC and other approaches in scenario with 7 edge servers, in which the coverage radius of edge server is $300m$ and that of cloud center is $400m$. (a) illustrates the average task execution time comparison; (b) illustrates the average energy cost per task; (c) illustrates the weighted reward for the objective function $-\frac{1}{T}\cdot\sum_{t=1}^{T}\omega_1 T_t+\omega_2\sum_{n=1}^{N}E_t^n$.

and the weighted reward of objective function (13) across different scenarios with varying numbers of UEs are illustrated in Fig. 5a, Fig. 5b and Fig. 5c, respectively. For the average task execution time, the proposed DTAC approach can significantly outperform both the TL and TL transformer approaches, as well as the HAPPO approach widely applied in MARL scenarios. In particular, though the centralized TAC achieves better performance compared to DTAC, it requires the global information for decisions, which cannot be applied in distributed scenarios. For the average energy cost per task, both the TL transformer and the proposed DTAC approaches demonstrate the lowest energy cost, comparable to that of the centralized TAC. Finally, since we aim at minimizing the objective function in (13), we also present the weighted reward $-\frac{1}{T}\cdot\sum_{t=1}^{T}\omega_1 T_t+\omega_2\sum_{n=1}^{N}E_t^n$ in Fig. 5c. First, the DTAC can achieve a similar performance as that of centralized TAC using only local information and distributed action. Second, concerning the increase in the number of UEs, only the performance of HAPPO exhibits unstable degradation, while the performance of other TL-based approaches degrade linearly. Due to the concatenated global state padded with zero element, conventional MARL approaches lose the position information, resulting in the weakness on the generalization ability in dynamic scenarios with varying numbers of agents.

Then, we consider a scenario with more edge servers and more UEs, which evaluates the ability of each approach to coordinate larger-scale UEs through a distributed manner. As presented in Fig. 6, DTAC also demonstrates a substantial reduction in task execution time and energy costs compared to other distributed approaches, including HAPPO and TL in this scenario with more UEs. In particular, TL using transformer can save more energy compared to DTAC at a cost with much worse task execution performance, which further demonstrates that the utilization of the learned critic model can significantly enhance the performance of decentralized TL. For the weighted reward for objective function, the performance difference among centralized TAC, DTAC and HAPPO is similar to that presented in Fig. 5c. In particular, the performance of both TL and TL using transformer approaches significantly degrades in this large scenario, highlighting the limitations of conventional TL methods in distilling a centralized policy for larger-scale agent deployment into distributed policies. Such

This article has been accepted for publication in IEEE Journal on Selected Areas in Communications. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JSAC.2025.3574611



(a) Average task execution time. (b) Average energy cost. (c) Average reward.

Fig. 7. Performance comparison between the proposed DTAC and other approaches in scenario with 7 edge servers and larger coverage, in which the coverage radius of edge server is $600m$ and that of cloud center is $800m$. (a) illustrates the average task execution time comparison; (b) illustrates the average energy cost per task; (c) illustrates the weighted reward for the objective function $-\frac{1}{T}\cdot\sum_{t=1}^{T}\omega_1 T_t+\omega_2\sum_{n=1}^{N}E_t^n$.

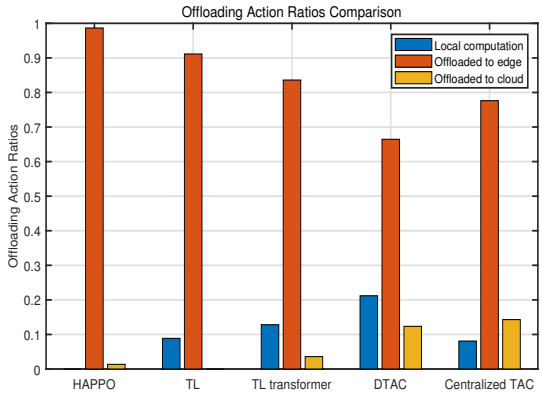


Fig. 8. Comparison on preference of each offloading action for different approaches in scenario of Fig. 6.

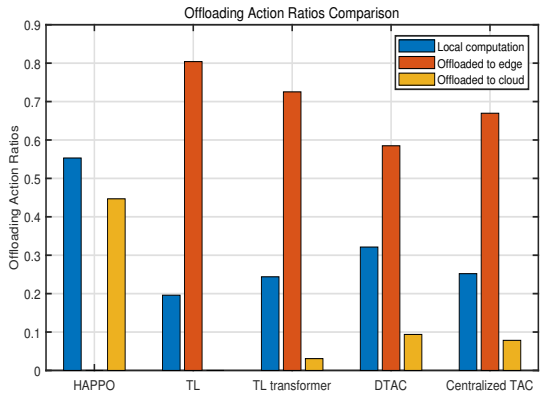


Fig. 9. Comparison on preference of each offloading action for different approaches in scenario of Fig. 7.

a result further presents the high-efficiency and stability of the proposed DTAC approach in scenarios comprising both small-scale and large-scale agents.

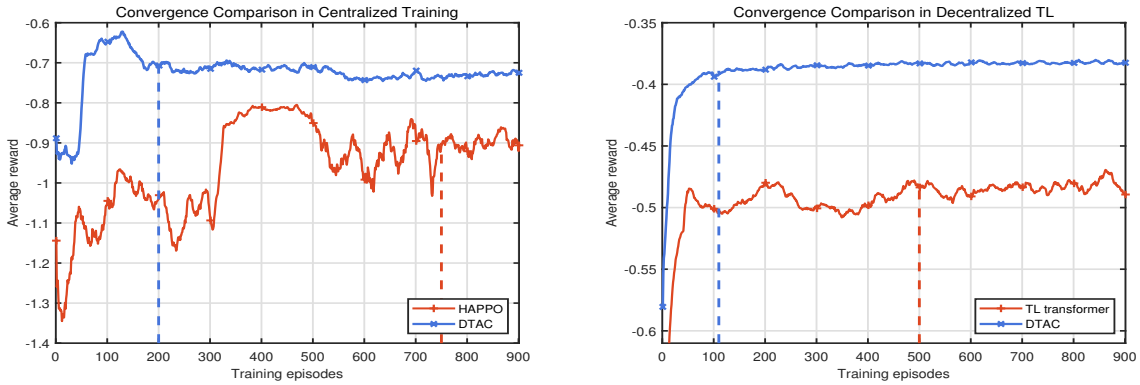
In previous scenarios, we mainly focused on a small cloud-edge computing network characterized by a limited coverage area for both the edge servers and the cloud center. To evaluate the DTAC in a larger cloud-edge computing network, we double the coverage radius of both edge servers and cloud center while keeping other setting as the same of Fig. 6. As presented in Fig. 7a, the DTAC continues to outperform other distributed approaches, while the HAPPO exhibits the poorest performance, even worse than that of the Random approach with the increase of number of UEs. Similarly, the proposed DTAC can also save most energy cost and obtains the best weighted reward, while the HAPPO achieves worst performance in both energy cost and weighted reward. Evidently, the significantly degraded performance of HAPPO can be attributed to the expanded coverage of edge servers and cloud center. With the larger coverage, both the transmission time and the required energy cost for offloading tasks significantly increase, leading to an increasing demand for distributed approaches to learn more intelligent policies that decides to offload or compute locally based on its position, task and resources status. However, the HAPPO fails to learn such intelligent offloading policies for all UEs, resulting in severely degraded performance in scenarios with larger coverage.

D. Action Comparison

TABLE III
ACTION SIMILARITY COMPARISON

Approaches \ Scenarios	Small area	Large area
TL transformer	0.103	0.028
DTAC	0.063	0.015

In this section, we further analyze the action preferences of each approach to evaluate their learning performance and discover the relationship between action preference and performance. Since the offloading action decides the position of task execution, it holds paramount importance for performance in cloud-edge computing networks. Consequently, we mainly focus on the offloading action preference in the following. We



(a) Centralized training (b) Decentralized TL

Fig. 10. The convergence comparison for different schemes in scenarios presented in Fig. 7 of the revised version.

present the offloading action preferences in scenarios of Fig. 6 and Fig. 7 in Fig. 8 and Fig. 9, respectively. In Fig. 8, the HAPPO converges to the local optimum that offloads most of tasks to the edge servers instead of intelligently offloading or computing task locally, leading to degraded performance. For the TL, it also suffers from the similar problem as that of the HAPPO. On the contrary, the TL using transformer also learn a similar distributed resource management policies as that of TL, resulting in much similar performance presented in Fig. 6. For the proposed DTAC approach, it can learn the intelligent offloading policies to enhance the network performance using only local information, which has similar offloading preference to that of centralized TAC. Notably, the increased preference on local computation of DTAC may relate to the unavailable information of other agents. Since the execution time of offloaded tasks are significantly impacted by the offloading actions of other agents, reducing the preference on offloading tasks may improve the network performance in distributed scenarios.

Then, the offloading action preferences in larger area depicted in Fig. 7 are illustrated in Fig. 9 for comparison. For the HAPPO, instead of offloading most tasks to edge servers as presented in Fig. 8, it converges to either offloading to the cloud center or compute at the local UEs. As previously mentioned, the larger coverage area necessitates enhanced learning capabilities in distributed approaches for intelligent offloading or local computing, due to increased transmission times and energy costs. Consequently, HAPPO fails to reach the global optimum, resulting in significantly degraded performance as illustrated in Fig. 7. In contrast, DTAC can learn effective offloading policies similar to those of centralized TAC approaches using only local information, thereby achieving much better performance than other distributed methods. Notably, DTAC also emphasizes local computation preference, as shown in Fig. 9, demonstrating that reducing task offloading preference can enhance network performance in distributed scenarios.

In particular, Table III further illustrates the action similarity (using KL divergence) between the two best approaches in previous simulations, TL transformer and DTAC approaches, and that of the centralized TAC model. Compared to the TL transformer, the proposed DTAC model achieves better distributed resource management with higher similarity to the centralized approach in both small-scale and large-scale scenarios, resulting in enhanced performance.

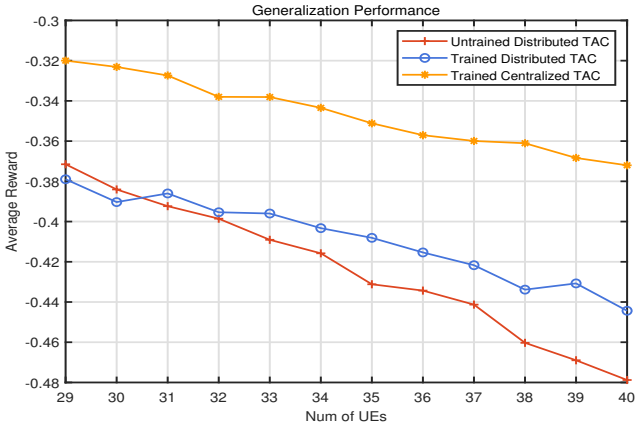


Fig. 11. The generalization evaluation of the DTAC approach involves applying the DTAC model, trained in scenarios with a smaller coverage area as depicted in Figure 6, to scenarios with a larger coverage area illustrated in Figure 7.

E. Generalization Performance

Moving forward, considering the adaptability requirement for ICPS, we evaluate whether the proposed DTAC can adapt to inexperienced scenarios. We apply the DTAC model trained in small-scale scenarios as depicted in Fig. 6 to large-scale scenarios as illustrated in Fig. 7. As Fig. 11 illustrates, compared to the trained DTAC model, the DTAC model inexperienced in this scenario can even achieve better performance when the number of UEs is less than 30. When the number of UEs increases, the DTAC model inexperienced in this scenario can also achieve a good performance, differing only marginally from that of the DTAC model trained in this scenario. Such a result demonstrates that the DTAC has sufficient generalization ability to adapt to diverse scenarios that are not even experienced during previous training.

F. Training Cost

TABLE IV
MODEL COMPLEXITY

Models	HAPPO Actor	HAPPO Critic	DTAC Actor	DTAC Critic
Flops	$N \cdot 577.4$k	$N \cdot 1038.5$k	$N \cdot 14.3$k	$N \cdot 14.5$k
Parameters	799 k	1436 k	8.17 k	8.52 k

Finally, considering ICPS deployment, we compare the training cost between the DTAC with other schemes. First, the overall training cost is divided into two components: the number of required training episodes and computational complexity. The convergence results for centralized training and decentralized TL are illustrated in Fig. 10a and Fig. 10b, respectively. In centralized training, the TAC model, utilizing the transformer architecture to efficiently capture correlations across UEs, demonstrates a significantly faster convergence rate and superior performance compared to HAPPO, saving 73% of training episodes. Note that Gaussian noise is introduced in DTAC during centralized training to enhance exploration of the state-action space, which may slightly reduce its performance compared to decentralized TL and testing. For decentralized TL, DTAC achieves more stable and efficient convergence than the TL transformer approach, saving 78% of training episodes. Since the only difference between DTAC and the TL transformer lies in the loss functions (26) and (22), this result further highlights that leveraging the converged critic model effectively guides the decentralized model toward better convergence.

For computational complexity, the floating point operations per second (Flops) and model size for conventional MLP-based HAPPO and transformer-based DTAC are summarized in Table IV, in which N denotes the number of UEs. Since TL transformer adopts the same model as DTAC and TL adopts the same model as HAPPO, we compare HAPPO and DTAC for instance. The number of parameters in HAPPO model is much larger than that of DTAC, especially for critic model that takes the concatenated form global state as input. The above simulation results demonstrate that the proposed DTAC requires substantially lower training costs compared to conventional HAPPO and TL schemes, making it more feasible for ICPS.

V. Conclusion

In this paper, considering the real-time requirements of ICPS, we propose the DTAC algorithm for distributed cloud-edge computing networks. The decentralized TL approach enables each UE learn to independently optimize task offloading and resource management while mitigating signaling overhead, leading to more efficient execution. To meet both performance and adaptability requirements, we first integrate the transformer with the actor-critic architecture in the centralized TAC model, effectively addressing the hybrid high-dimensional action space and dynamic scenarios with varying numbers of UEs. We then introduce a decentralized TL approach to transfer the centralized TAC model to the DTAC model. In the decentralized TL, the centralized critic model is used to guide the convergence of the DTAC model, enhancing its performance in distributed scenarios.

Simulation results demonstrate that the proposed DTAC can significantly outperform other MARL-based and TL-based schemes in both small-scale and large-scale scenarios. In inexperienced scenarios, DTAC also exhibits excellent performance, highlighting its strong generalization capability. Moreover, the proposed DTAC model and decentralized TL approach substantially reduce training costs by 73% compared to other MARL and TL schemes.

References

[1] K. Zhang, Y. Shi, S. Karnouskos, T. Sauter, H. Fang, and A. W. Colombo, "Advancements in Industrial Cyber-Physical Systems: An Overview and Perspectives," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 716–729, 2023.

[2] J. Chae, S. Lee, J. Jang, S. Hong, and K.-J. Park, "A Survey and Perspective on Industrial Cyber-Physical Systems (ICPS): From ICPS to AI-Augmented ICPS," *IEEE Transactions on Industrial Cyber-Physical Systems*, vol. 1, pp. 257–272, 2023.

[3] B. Dafflon, N. Moalla, and Y. Ouzrout, "The challenges, approaches, and used techniques of CPS for manufacturing in Industry 4.0: a literature review," *The International Journal of Advanced Manufacturing Technology*, vol. 113, pp. 2395–2412, 2021.

[4] E. Marilungo, A. Papetti, M. Germani, and M. Peruzzini, "From PSS to CPS design: a real industrial use case toward Industry 4.0," *Procedia Cirp*, vol. 64, pp. 357–362, 2017.

[5] C. Wittenberg, "Human-CPS Interaction-requirements and human-machine interaction methods for the Industry 4.0," *IFAC-PapersOnLine*, vol. 49, no. 19, pp. 420–425, 2016.

[6] A. Villalonga, G. Beruvides, F. Castaño, and R. E. Haber, "Cloud-Based Industrial Cyber–Physical System for Data-Driven Reasoning: A Review and Use Case on an Industry 4.0 Pilot Line," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 5975–5984, 2020.

[7] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, and S. Yin, "Industrial Cyberphysical Systems: A Backbone of the Fourth Industrial Revolution," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 6–16, 2017.

[8] J. Jin, K. Yu, J. Kua, N. Zhang, Z. Pang, and Q.-L. Han, "Cloud-Fog Automation: Vision, Enabling Technologies, and Future Research Directions," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, pp. 1039–1054, 2024.

[9] H. Ren, Y. Long, H. Ma, and H. Li, "Distributed Group Coordination of Random Communication Constrained Cyber-Physical Systems Using Cloud Edge Computing," *IEEE Transactions on Industrial Cyber-Physical Systems*, vol. 2, pp. 196–205, 2024.

[10] C.-C. Cheng, P.-C. Hsiu, T.-K. Hu, and T.-W. Kuo, "Oasis: A Mobile Cyber–Physical System for Accessible Location Exploration," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1744–1759, 2018.

[11] S. Deng, Z. Chen, F. Kuang, C. Yang, and W. Gui, "Optimal Control of Chilled Water System With Ensemble Learning and Cloud Edge Terminal Implementation," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7839–7848, 2021.

[12] K. Wang, J. Jin, Y. Yang, T. Zhang, A. Nallanathan, C. Tellambura, and B. Jabbari, "Task Offloading With Multi-Tier Computing Resources in Next Generation Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 306–319, 2023.

[13] 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.

[14] Li, Mushu and Cheng, Nan and Gao, Jie and Wang, Yinlu and Zhao, Lian and Shen, Xuemin, "Energy-Efficient UAV-Assisted Mobile Edge Computing: Resource Allocation and Trajectory Optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3424–3438, 2020.

[15] Zhou, Huan and Zhang, Zhenyu and Li, Dawei and Su, Zhou, "Joint Optimization of Computing Offloading and Service Caching in Edge Computing-Based Smart Grid," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1122–1132, 2023.

[16] S. Bi and Y. J. Zhang, "Computation Rate Maximization for Wireless Powered Mobile-Edge Computing With Binary Computation Offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[17] Lee, Gilsoo and Saad, Walid and Bennis, Mehdi, "An Online Optimization Framework for Distributed Fog Network Formation With Minimal Latency," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2244–2258, 2019.

[18] Bi, Suzhi and Huang, Liang and Zhang, Ying-Jun Angela, "Joint Optimization of Service Caching Placement and Computation Offloading in Mobile Edge Computing Systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4947–4963, 2020.

[19] Mao, Yuyi and Zhang, Jun and Song, S. H. and Letaief, Khaled B., "Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[20] Mach, Pavel and Becvar, Zdenek, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[21] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource Scheduling in Edge Computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[22] Zhou, Huan and Chen, Xin and He, Shibo and Zhu, Chunsheng and Leung, Victor C. M., "Freshness-Aware Seed Selection for Offloading Cellular Traffic Through Opportunistic Mobile Networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2658–2669, 2020.

[23] Zhou, Huan and Wu, Tong and Chen, Xin and He, Shibo and Wu, Jie, "RAIM: A Reverse Auction-Based Incentive Mechanism for Mobile Data Offloading Through Opportunistic Mobile Networks," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 6, pp. 3909–3921, 2022.

[24] Zhou, Huan and Jiang, Kai and Liu, Xuxun and Li, Xiuhua and Leung, Victor C. M., "Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1517–1530, 2022.

[25] Chen, Jiechen and Xing, Hong and Lin, Xiaohui and Nallanathan, Arumugam and Bi, Suzhi, "Joint Resource Allocation and Cache Placement for Location-Aware Multi-User Mobile-Edge Computing," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 25 698–25 714, 2022.

[26] Chen, Ying and Liu, Zhiyong and Zhang, Yongchao and Wu, Yuan and Chen, Xin and Zhao, Lian, "Deep Reinforcement Learning-Based Dynamic Resource Management for Mobile Edge Computing in Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4925–4934, 2021.

[27] Bi, Suzhi and Huang, Liang and Wang, Hui and Zhang, Ying-Jun Angela, "Lyapunov-Guided Deep Reinforcement Learning for Stable Online Computation Offloading in Mobile-Edge Computing Networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7519–7537, 2021.

[28] Peng, Haixia and Shen, Xuemin, "Deep Reinforcement Learning Based Resource Management for Multi-Access Edge Computing in Vehicular Networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2416–2428, 2020.

[29] He, Ying and Wang, Yuhang and Qiu, Chao and Lin, Qiuzhen and Li, Jianqiang and Ming, Zhong, "Blockchain-Based Edge Computing Resource Allocation in IoT: A Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2226–2237, 2021.

[30] X. Jiao, H. Ou, S. Chen, S. Guo, Y. Qu, C. Xiang, and J. Shang, "Deep Reinforcement Learning for Time-Energy Tradeoff Online Offloading in MEC-Enabled Industrial Internet of Things," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 6, pp. 3465–3479, 2023.

[31] Z. Zhou, Z. Wang, H. Yu, H. Liao, S. Mumtaz, L. Oliveira, and V. Frascolla, "Learning-Based URLLC-Aware Task Offloading for Internet of Health Things," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 396–410, 2021.

[32] N. Gholipour, M. D. de Assuncao, P. Agarwal, J. Gascon-Samson, and R. Buyya, "TPTO: A Transformer-PPO based Task Offloading Solution for Edge Computing Environments," in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, 2023, pp. 1115–1122.

[33] M. Han, X. Sun, X. Wang, W. Zhan, and X. Chen, "Joint Caching, Communication, Computation Resource Management in Mobile-Edge Computing Networks," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, 2024, pp. 1–6.

[34] Z. Gao, L. Yang, and Y. Dai, "Fast Adaptive Task Offloading and Resource Allocation via Multiagent Reinforcement Learning in Heterogeneous Vehicular Fog Computing," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6818–6835, 2023.

[35] S. Ozer, H. E. Ilhan, M. A. Ozkanoglu, and H. A. Cirpan, "Offloading Deep Learning Powered Vision Tasks From UAV to 5G Edge Server With Denoising," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 8035–8048, 2023.

[36] H. Huang, W. Zhan, G. Min, Z. Duan, and K. Peng, "Mobility-Aware Computation Offloading With Load Balancing in Smart City Networks Using MEC Federation," *IEEE Transactions on Mobile Computing*, vol. 23, no. 11, pp. 10 411–10 428, 2024.

[37] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. P. Adams, and S. Levine, "Why Generalization in RL is Difficult: Epistemic Pomdps and Implicit Partial Observability," *Advances in neural information processing systems*, vol. 34, pp. 25 502–25 515, 2021.

[38] 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.

[39] M. Han, Z. Chen, and X. Sun, "Multiple Access via Curriculum Multi-Task HAPPO Based in Dynamic Heterogeneous Wireless Network," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[40] T. Phan, F. Ritz, P. Altmann, M. Zorn, J. Nüßlein, M. Kölle, T. Gabor, and C. Linnhoff-Popien, "Attention-based recurrence for multi-agent reinforcement learning under stochastic partial observability," in *International Conference on Machine Learning*. PMLR, 2023, pp. 27 840–27 853.

[41] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, pp. 1–40, 2016.

[42] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer Learning Promotes 6G Wireless Communications: Recent Advances and Future Challenges," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 790–807, 2021.

[43] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer Learning in Deep Reinforcement Learning: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 344–13 362, 2023.

[44] K. Shao, Y. Zhu, and D. Zhao, "Starcraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 1, pp. 73–84, 2018.

[45] Xing, Hong and Cui, Jingjing and Deng, Yansha and Nallanathan, Arumugam, "Energy-Efficient Proactive Caching for Fog Computing with Correlated Task Arrivals," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2019, pp. 1–5.

[46] Li, Liying and Zhao, Guodong and Blum, Rick S., "A Survey of Caching Techniques in Cellular Networks: Research Issues and Challenges in Content Placement and Delivery Strategies," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 1710–1732, 2018.

[47] C. Liu, X. Xu, and D. Hu, "Multiobjective Reinforcement Learning: A Comprehensive Overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2015.

[48] T. Lattimore, M. Hutter, and P. Sunehag, "The Sample-complexity of General Reinforcement Learning," in *International Conference on Machine Learning*. PMLR, 2013, pp. 28–36.

[49] Y. Zhao, G. Wang, C. Tang, C. Luo, W. Zeng, and Z.-J. Zha, "A battle of network structures: An empirical study of cnn, transformer, and mlp," *arXiv preprint arXiv:2108.13002*, 2021.

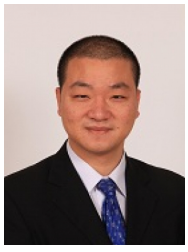
[50] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.

Mingqi Han received the B.Eng. degree in Communication Engineering from School of Electronics and Communication Engineering, Shenzhen Campus of Sun Yat-sen University, Shenzhen, China in 2022. He is currently pursing for the M.E. degree in Information and Communication Engineering with School of Electronics and Communication Engineering, Shenzhen Campus of Sun Yat-sen University, Shenzhen, China. His research interests include machine learning, reinforcement learning and multiple access.

Xinghua Sun (M'13) received the B.S. degree from Nanjing University of Posts and Telecommunications (NJUPT), China, in 2008 and the Ph.D. degree from City University of Hong Kong (CityU), China, in 2013. In 2010, he was a visiting student with National Institute for Research in Digital Science and Technology (INRIA), France. In 2013, he was a postdoctoral fellow at CityU. From 2015 to 2016, he was a postdoctoral fellow at University of British Columbia, Canada. From July to Aug. 2019, he was a visiting scholar at Singapore University of Technology and Design, Singapore. From 2014 to 2018, he was an associate professor with NJUPT. Since 2018, he has been an associate professor with Sun Yat-sen University, Guangdong, China. His research interests are in the area of stochastic modeling of wireless networks and machine learning for next generation wireless communications and networks. He was a co-recipient of the Best Paper Award from the EAI IoTaaS in 2023 and the IEEE FCN in 2024.

Xijun Wang (Member, IEEE) received the B.S. degree (Hons.) in communications engineering from Xidian University, Xi'an, Shaanxi, China, in 2005, and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in January 2012. He was an Assistant Professor from 2012 to 2015 and an Associate Professor from 2015 to 2018 with the School of Telecommunications Engineering, Xidian University. From 2015 to 2016, he was a Research Fellow with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design. He is currently an Associate Professor with Sun Yat-sen University. He has published several IEEE journals and conference proceedings in the areas of wireless networks and patents related to heterogeneous networks. His current research interests include UAV communications, edge computing, and age of information. He served as the technical program committee member for numerous IEEE conferences. He was a recipient of the Best Paper Award from the IEEE ICCC 2013. He also served as the Publicity Chair for IEEE ICCC 2013 and the Technical Program Co-Chair for the Wireless Communications Systems Symposium and IEEE ICCC 2016. He is currently a reviewer for several IEEE journals. He was recognized as an Exemplary Reviewer of the IEEE WIRELESS COMMUNICATIONS LETTERS in 2014. He is currently an Associate Editor of IEEE ACCESS.

Wen Zhan (Member, IEEE) received the B.S. and M.S. degrees from the University of Electronic Science and Technology of China, China, in 2012 and 2015, respectively, and the Ph.D. degree from the City University of Hong Kong, China, in 2019. He was a Research Assistant and a Postdoctoral Fellow with the City University of Hong Kong. Since 2020, He has been with the School of Electronics and Communication Engineering, Sun Yat-sen University, China, where he is currently an Assistant Professor. His research interests include Internet of Things, modeling, and performance optimization of next-generation mobile communication systems.



Xiang Chen (Member, IEEE) received the B.E. and Ph.D. degrees from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2002 and 2008, respectively. From July 2008 to December 2014, he was with the Wireless and Mobile Communication Technology Research and Development Center (Wireless Center) and the Aerospace Center, Tsinghua University. In July 2005 and from September 2006 to April 2007, he visited NTT DoCoMo Research and Development (YRP), and Wireless Communications and Signal Processing (WCSP) Laboratory, National Tsing Hua University. Since January 2015, he has been with the School of Electronics and Information Technology, Sun Yat-sen University, where he is currently a Full Professor. His research interests mainly focus on 5G/6G wireless and mobile communication net works, and the Internet of Things (IoT).