Multi-Agent Deep Reinforcement Learning for Resource Scheduling in Hybrid-Energy MEC Systems Under Long-Term Constraints

QingHua Zhu, *Senior Member, IEEE*, JinYu Liu, Yan Hou, DaWen Lai, and ZiMing Ou

***Abstract*—In a mobile edge computing environment where mobile devices are powered by green energy, minimizing power consumption and latency under long-term MEC energy constraints faces several challenges: unpredictable tasks, keeping both the privacy of the mobile device's battery and efficient task scheduling, stabilization of power consumption and latency accumulation, and non-convex optimization for a continuous-discrete decision space. To tackle these challenges, we investigate the latency-aware resource-constrained scheduling (LARCS) problem in a hybrid-energy edge-cloud MEC system to minimize energy consumption and latency under long-term MEC energy constraints. For such a scenario, for the first time, we address protecting battery privacy and mitigating energy rebound peaks, which indicate sudden power surges from concurrent high-load tasks. We model the LARCS using mixed integer nonlinear programming (MINLP) and prove its NP-completeness. Then, it is reformulated as a Markov game. We adopt the multi-actor-attention-critic learning mechanism to preserve user privacy and employ a reward shape in reward functions for mitigating energy rebound peaks. On the above basis, we propose a latency-aware resource-constrained scheduling algorithm on the basis of multi-agent deep reinforcement learning (LARC-MADRL) to coordinate optimizations for multiple mobile users. This proposed algorithm requires no prior knowledge of uncertain parameters, operates independently of hybrid-energy dynamic models, and maintains user privacy. Numerical simulations validate that our proposed algorithm outperforms benchmark algorithms in minimizing power consumption, latency, and dropout rate while achieving long-term benefits, better convergence, improved stability, and enhanced scalability.**

***Note to Practitioners*—In mobile edge computing systems, such as some kinds of Internet-of-Things applications powered by renewable energy, a key practical problem is to efficiently manage task offloading to minimize both power consumption and latency without violating long-term energy constraints. Such a scenario is particularly challenging due to unpredictable workloads, the requirement to protect device battery privacy, and the risk of sudden power surges from concurrent tasks. This work addresses such issues and provides a scalable scheduling algorithm based on multi-agent deep reinforcement learning, significantly improving system efficiency and reliability. This work contributes a novel, model-free approach that enables mobile devices to make intelligent, decentralized offloading decisions in real time, requiring no complex network modeling.**

Q.H. Zhu, J.Y. Liu,Y. Hou, D.W. Lai, and Z.M. Ou are with the School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China. E-mail: zhuqh@gdut.edu.cn, 3122004620@mail2.gdut.edu.cn, houyan@gdut.edu.cn, 2112405046@mail2.gdut.edu.cn, 3122004623@mail2.gdut.edu.cn. Corresponding author: Qinghua Zhu (e-mail: zhuqh@gdut.edu.cn).

This work was supported in part by the National Undergraduate Innovation Training Program under Grant 202511845050 and in part by the Natural Science Foundation of Guangdong Province, China, under Grant 2026A1515012510.

***Index Terms*—Deep reinforcement learning, green energy, mobile edge computing, task offloading.**

I. INTRODUCTION

With the exponential increase in mobile communication and Internet of Things (IoT) technologies, there has been a significant rise in the demand for computational power. This surge is further exacerbated by the advent of computation-intensive applications, such as edge-assisted autonomous driving and mobile augmented reality. However, mobile devices (MDs) are hindered by their limited computing capacity and available energy, which poses a significant challenge in managing these compute-intensive tasks. Consequently, cloud computing has emerged as a viable solution to furnish the necessary computational resources and storage [1]. Nonetheless, the geographical distance between cloud servers and mobile devices can result in prolonged latency, which poses a challenge in meeting users' real-time requirements.

To address the aforementioned challenge, mobile edge computing (MEC) facilitates localized data processing and mitigates servers the necessity for data transmission to remote servers. This approach leads to expedited response time, enhanced user experiences, and alleviated network congestion [2]. Furthermore, MEC retains the energy efficiency advantages of cloud computing by allowing devices to offload energy-intensive tasks to the MEC servers (MECSs), thereby conserving battery life. Additionally, MEC consumes less energy than cloud computing and reduces carbon emissions [3].

However, resource scheduling, green energy harvesting, and storage in MEC networks to minimize power consumption and latency present significant challenges due to their strong dynamics and uncertainty [4]. On the one hand, in the time-varying network environment, tasks are generated randomly on MDs and vary in size, computational complexity, and strictness of their deadline requirements, all of which must be met for the quality of service (QoS) demands of end-users, such as low latency and rapid mobility. Moreover, the unpredictability is exacerbated by imperfect channel state information, which introduces a higher likelihood of transmission failures. On the other hand, hybrid-energy MEC systems, which rely on a combination of sustainable energy (e.g., solar or wind power) and grid power (e.g., fossil fuels), face significant challenges due to the unpredictable characteristics of green energy. The supply of sustainable energy is highly susceptible to fluctuations caused by changing environmental conditions, leading to

instability in energy harvesting. This unpredictability results in intermittent availability of green energy, which poses a risk of power outages if not managed properly. Moreover, while stored energy can be used as a backup, it comes with higher operational costs and environmental impact, making it less desirable as a primary energy source. Consequently, there is an urgent need for advanced resource scheduling methods in hybrid-energy MEC systems to efficiently balance energy use, minimize task completion times, and adapt to dynamic conditions.

Traditional resource scheduling methods in MEC systems, such as heuristic algorithms [5], approximate algorithms [6], and convex optimization techniques [7], can achieve good results under general conditions but are less effective in complex, real-world MEC environments — especially hybrid-energy MEC scenarios. First, they cannot adapt to the stochasticity of green energy (solar/wind) supply, as fixed, pre-defined heuristic rules or convex optimization's reliance on prior system dynamics fail to handle real-time energy fluctuations. Second, they prioritize single-slot optimization that immediate latency/energy reduction without considering inter-temporal balance and ignore long-term energy constraints that require coordinating decisions across slots to avoid energy depletion in low-harvesting periods—critical for hybrid-energy system stability. This leads to short-sighted decisions (e.g., overusing harvested energy in peak periods). Third, energy rebound peaks are severe: concurrent high-load task offloading can push mobile devices' power consumption to a high value, exceeding mobile devices' hybrid-energy supply capacity, causing service disruptions, accelerating hardware fatigue, and exacerbating energy deficits.

Furthermore, while long-term average performance, which often requires complete future information, is more convincing than single time slot performance, traditional methods struggle to accurately predict future outcomes in dynamic systems. In contrast, reinforcement learning (RL) methods, such as deep Q-network (DQN) [8], deep deterministic policy gradient (DDPG) [9], and multi-agent DDPG (MADDPG) [10], offer a model-free approach without prior knowledge of the environment. These methods can optimize long-term average rewards through training on historical data, autonomously learn system dynamics, and make appropriate decisions in real time, making them particularly well-suited for the uncertain conditions of hybrid-energy MEC systems. Therefore, we prefer RL approaches to effectively manage resource scheduling in such complex environments. However, while previous RL methods like DQN, DDPG, and MADDPG perform well in dynamic MEC environments, they suffer from exponential increases in computational cost and execution time as the number of MDs grows. Thus, to overcome these limitations, a flexible and scalable approach to resource scheduling is desired to efficiently manage computational demands as agent numbers increase, without compromising performance.

If too much energy is consumed in a certain time slot, it may lead to insufficient energy in subsequent time slot, affecting the system's ability to continue operating. However, traditional single-slot optimization always overlooks long-term energy balance, resulting in the system being unable to provide services in the later stages. Such requirements are referred to as long-term MEC energy constraints. Thus, we investigate a latency-sensitive resource-constrained scheduling (LARCS) problem for an edge-cloud collaboration MEC system to promote the end user's quality of experience (QoE) with the consideration of long-term MEC energy constraints and the aim of minimizing overall energy consumption and latency.

However, several challenges must be addressed to achieve this goal:

1) It is challenging to keep both the privacy of the mobile device's battery and efficient task scheduling. Efficient task scheduling in hybrid-energy systems often depends on the battery level and energy harvesting rate. However, battery levels can reveal user behavior patterns like daily routines and locations. Efficient scheduling needs transparency, but privacy-preserving schemes usually restrict access to battery information.
2) It is difficult to stabilize a distributed MEC system with long-term MEC energy constraints, which involve energy time-coupling and latency accumulation. If not properly managed, excessive energy consumption can lead to future power shortages, disrupting task execution, while uncontrolled latency accumulation degrades service quality. Furthermore, dynamic load fluctuations may cause server overloads and instability, especially when multiple MDs trigger high-power computation or data transmission simultaneously, resulting in rebound peaks. Addressing these challenges is crucial to ensuring energy-efficient, low-latency, and stable MEC operations.
3) It is challenging to handle unpredictable tasks in a heterogeneous environment. The unpredictable nature of mobile devices' workloads, computing requirements, real-time tasks, rapid mobility, and stochastic parameters (e.g., the supply of green energy and the dynamic wireless channel) introduce significant complexity to MEC systems. These uncertainties demand adaptive strategies that can efficiently allocate resources while maintaining system stability.

Given these challenges, existing model-based [12]-[14], [23]-[27], [29], [30], [32] and model-free [15]-[22], [28], [31] methods are not suitable for our problem due to the limitations of traditional approaches and conventional RL methods mentioned above. In this paper, we address non-divisible and latency-sensitive tasks within a heterogeneous and dynamic MEC environment, formulating a latency-aware resource-constrained scheduling problem to minimize overall latency and energy consumption while satisfying long-term MEC energy constraints. To this end, we propose a latency-aware resource-constrained scheduling algorithm by employing multi-agent deep reinforcement learning (LARC-MADRL) with an attention mechanism [11] to support scalable and flexible coordination for multiple agents.

We make the major contributions as follows:

1) We investigate an hybrid-energy MEC network architecture consisting of multiple MDs with energy harvesting capabilities, multiple base stations integrated with edge

servers, and a central cloud server. As far as we are aware, no prior research has addressed both the battery privacy of MDs and the impact of battery rebound peaks in MEC systems. Specifically, battery privacy is preserved using an attention-based encoded mechanism.

2) By taking into consideration the heterogeneous and dynamic MEC environment, the mobility of MDs, and the supply of green energy, we first formulate an LARCS problem subject to long-term MEC energy constraints, aiming to minimize energy consumption and latency to enhance the end-user's QoE for an edge-cloud collaboration MEC system in the form of mixed integer non-linear programming (MINLP) which is proved to be NP-completeness. Then, we further reformulate it as a Markov game whose environment state, action, and normalized reward function are well defined. As far as we know, this is the first work that explicitly incorporates long-term MEC energy constraints, including both long-term energy and latency considerations, into resource scheduling for hybrid-energy MEC systems.
3) Within the Markov game framework, we introduce a novel reward shaping mechanism that serves as a task penalty and energy deficit indicator, effectively guiding real-time energy consumption in the MEC system. This mechanism not only decouples long-term MEC energy constraints but also allows for adaptive management of resources based on current system states.
4) Within the addressed the Markov game, we propose a novel, scalable, latency-aware resource-constrained scheduling algorithm LARC-MADRL with an attention mechanism. Notably, we propose a model-free algorithm without prior knowledge of the uncertain MEC systems; it can automatically learn and accurately evaluate the LARCS decisions, thus handling dynamic and diverse situations. Additionally, the attention mechanism renders every agent to selectively incorporate relevant information from other agents during training, offering advantages in terms of generalization capability, scalability, and computational overhead compared to previous methods, while also preserving user privacy.

Therefore, we summarize the novelties for this work as follows. To preserve mobile user privacy, we introduce a decentralized coordination mechanism based on a multi-actor attention-critic (MAAC) architecture. Our method aggregates agent information via a weighted attention mechanism, reducing the critic's input dimensionality from linear scaling to a fixed-size embedding while preserving user privacy by sharing only latent feature vectors. For mixed-integer nonlinear programming with long-term energy constraints, we propose a novel reward-shaping formulation that serves as a soft-constraint proxy to satisfy long-term stability constraints without an explicit model. We organize the paper as follows. Section II reviews the relevant literature. Section III outlines the system model and presents the problem formulation. Section IV gives a scheduling algorithm based on MADRL. Section V presents experimental results, comparing the advantages of the proposed algorithm with other scheduling approaches. Section VI concludes this work.

II. RELATED WORK

This section summarizes existing work relevant to resource scheduling in edge computing and the optimization of energy consumption and task response time in MEC systems.

Resource scheduling in edge computing. Resource scheduling in edge computing receives considerable attention from academia [12]-[22]. Mohajer *et al.* [12] propose a fairness-aware model to minimize energy use in ultra-dense heterogeneous networks with cross-tier interference. Jiang *et al.* [13] optimize computation and bandwidth allocation in MEC offloading to improve user quality of experience (QoE). Zhang *et al.* [14] design a task scheduling and container strategy for edge servers with multiple processors. Daghayeghi *et al.* [15] introduce an MADRL framework within the context of a single-cell non-orthogonal multiple access-assisted MEC environment to minimize task completion time and consumed energy, considering cooperation between MDs to adjust their strategies. Bi *et al.* [16] combine Lyapunov optimization with DRL to improve MEC offloading under queue and power constraints. Gebrekidan *et al.* [17] introduce a novel algorithm for task offloading in an MEC environment, which addresses the challenges posed by diverse constraints at user devices (UDs), the wireless network, and the server. This approach involves deploying client agents at UDs to manage resource allocation decisions, alongside a master agent at the server that formulates combinatorial decisions informed by the client actions. Wang *et al.* [18] develop a DQN-based scheme to optimize base station power and unmanned aerial vehicle service zones for maximizing spectrum efficiency. Sun *et al.* [19] propose a multi-branch DQN to optimize offloading and resource allocation in multi-user MEC systems. Chen *et al.* [48] focus on joint task scheduling and energy management to maximize the utility of MEC systems with hybrid energy supply. Unlike prior work [12], [27], [30]-[31], [48], which emphasizes energy harvesting at base stations, this study considers energy harvesting at mobile devices. A notable gap in these earlier hybrid-energy MEC studies [12], [15]-[17], [21], [27], [30]-[31], [48] is their lack of attention to two critical practical issues: energy rebound peaks (i.e., sudden power surges caused by concurrent high-load tasks) and potential privacy leakage associated with the mobile-device battery status. These overlooked aspects not only affect the stability and efficiency of MEC systems but also pose risks to user privacy, highlighting the requirements for more comprehensive optimization frameworks to address these challenges together.

However, while the aforementioned studies primarily focus on an MEC system with a flat or two-tiered architecture, our study investigates the resource scheduling problem in a more complex cloud-edge-device architecture, where resources are dynamically allocated across different layers to optimize performance and reduce latency. Furthermore, while prior centralized training and decentralized execution (CTDE) MADRL methods [15], [17], [22] suffer from two key limitations: 1) high communication overhead between agents; 2) heightened user privacy concerns, as individuals are generally reluctant

to disclose sensitive personal information (e.g., battery status, task data, user preferences, or network conditions), our proposed method alleviates these limitations through an attention mechanism and a shared critic structure.

Optimization of energy consumption and task response time in MEC systems. In the past few years, significant research efforts focus on the optimization of consumed energy [26]-[27], [29]-[32] and task response time [23]-[30] in an MEC system. Fan *et al.* [23] propose a scheme to minimize task processing latencys in an edge-cloud environment with edge-to-edge, multi-service, and multi-task cooperations. Ma *et al.* [24] address the computation-communication latency, edge node heterogeneity, and task arrival dynamics trade-off in their dynamic scheduling approach. Chen *et al.* [25] minimize latency by using a convex approximation technique to address non-convexity and an online algorithm to prevent cache violations in MEC systems. Tuli *et al.* [26] develop an extended gradient-based optimization to minimize consumed energy and response time, optimizing QoS. Ma *et al.* [27] focus on minimizing costs by considering latency, energy use, and cloud fees in green multi-tier edge computing. Liu *et al.* [28] propose a joint optimization for multi-resource management and task offloading to maximize system utility, leveraging service migration and vehicle resources. Jiang *et al.* [29] propose a hybrid adaptive algorithm to maximize task offloading utility which improves communication costs, energy consumption, and task completion time. Perin *et al.* [30] focus on executing computing jobs generated by access nodes to minimize energy purchases. Gu *et al.* [31] aim to reduce long-term operational expenditure by formulating the problem as a mixed-integer linear program. Xue *et al.* [32] aim to minimize the combined overhead of computing resource costs and consumed energy for all task request users in an MEC and device-to-device communication network.

However, unlike the above studies, we integrate energy harvesting technologies and impose long-term MEC energy constraints in resource scheduling to maintain system stability, as failure to meet these constraints can lead to suboptimal performance in practice. To the authors' best knowledge, no existing work optimizes both consumed energy and task response time in an hybrid-energy MEC system to enhance the end-user's QoE while simultaneously considering the long-term MEC energy constraints as addressed in this paper.

To summarize, as listed in Table I, although several model-based [13], [23], [24], [27] and model-free [15], [16], [22], [31] resource scheduling methods have been proposed to optimize consumed energy, latency, or both for long-term benefits in MEC environments, they face challenges arising from real-time and non-divisible tasks, edge–cloud collaboration, user mobility, unpredictable green energy supply, adaptive QoS requirements, and the requirement to adapt to heterogeneous and dynamic conditions, making it challenging for existing methods to provide a robust and comprehensive solution. In contrast, the resource scheduling approach proposed in this work demonstrates significant advancements by exhibiting superior generalization capability, scalability, and computational efficiency compared to prior work. Moreover, a significant aspect of our work is its pioneering focus on user battery privacy in hybrid-energy MEC systems within resource scheduling algorithms. To our knowledge, the existing approaches [12]-[32], [49] have not effectively incorporated privacy protection measures. This research is the first to integrate an attention mechanism specifically designed to safeguard user battery data while optimizing scheduling performance. This advancement not only enhances the robustness of the algorithm but also addresses a critical concern in MEC environments, setting a new precedent for future research in the field.

III. SYSTEM MODELS AND PROBLEM FORMULATION

This section provides an overview of the system and discusses the long-term MEC energy constraints. Subsequently, we describe the system models, including the computation, communication, and energy harvesting models. Finally, we formulate LARCS and its variants.

A. Hybrid-energy MEC Systems

We consider a system that operates in discrete time slots, denoted by $t \in \mathcal{T} = \{0, 1, \cdots, T-1\}$, where each time slot has a duration of Δt seconds. As illustrated in Fig. 1, an hybrid-energy MEC system consisting of the following three layers: MDs, edge servers, and cloud server.

MD layer:

Let N stand for the number of MDs and $\mathcal{N} = \{1, 2, \ldots, N\}$ for the index set of MDs. Each MD_i, where $i \in \mathcal{N}$, generates real-time tasks for each time slot, each with specific deadlines. These tasks are described as a triplet $H_n^t = (s_n^t, c_n^t, \tau_n^t)$, where s_n^t stands for the task data size, c_n^t denotes the required computation resources, and τ_n^t stands for the latency constraints. To optimize energy usage and enhance sustainability, each MD is equipped with an energy harvester that collects green energy sources, e.g., solar and wind, and stores the harvested energy in its battery [15], [17], [36]. This setup enables the MDs to operate more efficiently by utilizing renewable energy for task processing, thereby reducing reliance on the conventional power grid. During a given time slot, the devices are supposed to be still, and the wireless channels are considered stable. However, due to the mobility of the MDs, their locations may vary across different time slots.

Edge server layer: Let M stand for the number of edge servers and $\mathcal{M} = \{1, 2, \ldots, M\}$ for the index set of edge servers. Let MECS_m denote edge server m with $m \in \mathcal{M}$. Let F_m denote the computation capacity of MECS_m with $m \in \mathcal{M}$. Each base station is configured with an edge server with the capacity of cloud and information technology services. Every edge server may serve a variety of MDs across time slots due to their mobility, processing requests from various MDs and providing necessary computation capabilities and resorces at the base station. For latency-sensitive tasks, the limited resources of MDs may not suffice to meet the QoE requirements; therefore, tasks on the MDs will be offloaded to nearby edge servers to reduce computation latency and local energy consumption.

Cloud Server Layer: Let F_c denote the computation capacity of the cloud server. Cloud computing handles latency-tolerant

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. 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/TASE.2026.3690812

TABLE I
COMPARISON OF RELATED WORK(✓ INDICATES THE PRESENCE OF THE FEATURE)

Work	Environment			Task Type		Method	Optimization Objectives			Features	
	Edge-Cloud	Mobility	Green Energy	Real-Time	Indivisible	RL	Energy Consumption	Latency	Long-Term Benefits	Battery	Privacy
[12]			✓		✓		✓				
[13]		✓		✓	✓		✓	✓	✓		
[14]				✓				✓			
[15]			✓	✓		✓	✓	✓	✓		
[16]			✓	✓	✓	✓	✓		✓		
[17]			✓	✓	✓	✓	✓	✓			
[18]		✓		✓		✓		✓			
[19]				✓	✓	✓		✓			
[20]		✓	✓	✓	✓	✓		✓			
[21]		✓	✓	✓	✓	✓		✓			
[22]		✓		✓	✓	✓	✓	✓	✓		
[23]	✓			✓	✓			✓	✓		
[24]	✓	✓		✓	✓			✓	✓		
[25]	✓	✓		✓	✓			✓			
[26]	✓			✓			✓	✓			
[27]	✓	✓	✓	✓	✓		✓	✓	✓		
[28]	✓	✓		✓	✓	✓		✓			
[29]				✓	✓		✓	✓			
[30]			✓	✓	✓		✓	✓			
[31]			✓	✓	✓	✓	✓		✓		
[32]				✓	✓		✓				
[48]		✓	✓	✓			✓		✓		
This work	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	

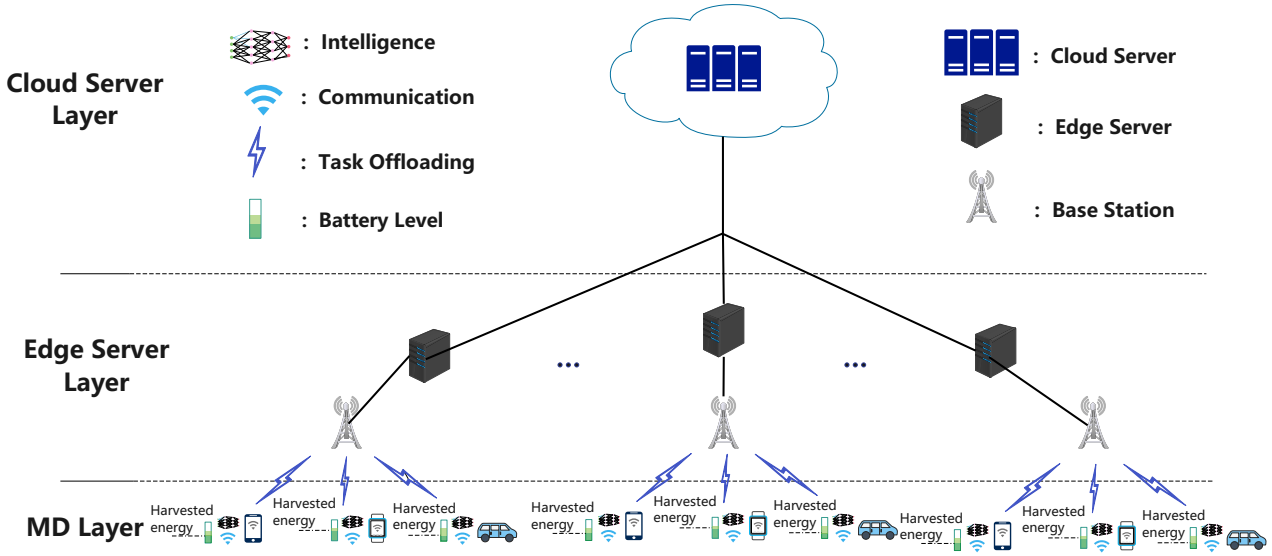

Fig. 1. The hybrid-energy MEC system consists of a cloud server, a number of base stations (BSs) with MECSs, and MDs equipped with energy harvesters. Each MD can either process its tasks locally or offload them to a nearby edge server, depending on the allocated computation and communication resources.

applications based on the actual deployment locations of MDs, while edge computing focuses on latency-aware services. When the capacity of edge servers is insufficient to handle the demands of MDs, tasks are offloaded to cloud servers, which provide essential support.

Next, we will introduce the long-term MEC energy constraints in hybrid-energy MEC systems.

1) Task offloading constraint: Let $\mathcal{M}' = \{0, 1, \ldots, M\}$ represent the index set of computation locations, where $\mathcal{M}'_0$ indicates the local computation and $\mathcal{M}'_m$ indicates the computation on MECS$_m$ with $m \in \mathcal{M}$. For each time slot $t \in \mathcal{T}$, let a^t_{nm} denote whether MD$_n$ offloads its task to MECS$_m$ ($m \in \mathcal{M}$) or local computing if $m = 0$. We suppose that every task is indivisible and must be fully computed locally or fully offloaded to one server. Thus, the MDs offloading decisions must satisfy:

$$a^t_{nm} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}', t \in \mathcal{T}. \tag{1}$$

and

$$\sum_{m=0}^{M} a_{nm}^{t} = 1, \quad \forall n \in \mathcal{N}, t \in \mathcal{T}. \tag{2}$$

Then, we introduce a binary indicator x_{nm}^{t}, where $x_{nm}^{t} = 1$ indicates that the resource-limited MECS$_m$ cannot handle the latency-tolerant tasks, and the tasks offloaded from MD$_n$ is sent to the cloud server for further processing. Conversely, $x_{nm}^{t} = 0$ means that the tasks are executed on MECS$_m$, indicating that this MECS$_m$ has sufficient resources to handle the stochastic tasks generated by MDs. Thus, the MECSs offloading decisions must satisfy:

$$x_{nm}^{t} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{3}$$

2) Computation resource allocation constraint: Each MECS must allocate a positive computing resource value to every MD, and the cloud server must allocate a positive computing resource value to every MECS. Thus, we have

$$f_m \geq 0, \quad \forall m \in \mathcal{M}. \tag{4}$$

$$f_{nm} \geq 0, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}. \tag{5}$$

The sum of computing resources assigned to the MDs must not exceed the computing capacity of their respective MECS. Similarly, the sum of computing resources allocated to the MECS must not exceed the computing capacity of the cloud server. Thus, we have

$$\sum_{m=1}^{M} f_m^{t} \leq F_c, \quad \forall m \in \mathcal{M}, t \in \mathcal{T}. \tag{6}$$

$$\sum_{n=1}^{N} f_{nm}^{t} \leq F_m, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{7}$$

3) Battery level constraint: For $\forall t \in \mathcal{T}$, let b_n^t represent the battery status of MD$_n$. The battery status must remain within defined limits, specifically not falling below the minimum threshold $b^{\min}$ and exceeding the maximum threshold b^{max}. Thus, we establish the constraint:

$$b^{\min} \leq b_n^{t} \leq b^{max}, \quad \forall n \in \mathcal{N}, t \in \mathcal{T}. \tag{8}$$

The change in the battery level is computed using (26). If the battery level at time $t+1$ falls below $b^{\min}$, the task is aborted. Notably, the MD's battery status directly influences its task offloading strategy. For instance, when the battery level is too low to support task offloading at the current transmission power, the MD is adjusted by lowering the transmission power accordingly.

For realistic consideration, renewable energy sources are subject to change and often inadequate to fully support telecommunication network demands. Thus, we consider the hybrid-energy MEC system by integrating renewable energy sources with the power grid. We define e_n^t as the harvested green energy available for exceuting tasks on MD$_n$. Then, we require that:

$$e_n^{t} \geq 0, \quad \forall n \in \mathcal{N}, t \in \mathcal{T}. \tag{9}$$

If an MD cannot make full use of the harvested green energy, the virtual energy may accumulate and potentially grow indefinitely over time. According to [13] and [16], to maintain stability, we must ensure the following condition is satisfied:

$$\lim_{T \to +\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left\{b_n^{t}\right\} < +\infty, \quad \forall n \in \mathcal{N}. \tag{10}$$

TABLE II
LIST OF KEY NOTATIONS

Notation	Definition
N	Number of MDs
M	Number of MECSs
a_{nm}^t	1 if MD$_n$ offloads to MECS$_m$ in time slot t, 0 otherwise
x_{nm}^t	1 if MECS$_m$ forwards a task to the cloud in time slot t, 0 otherwise
s_n^t	Task data size in time slot t
c_n^t	Required computation resources in time slot t
τ_n^t	Latency constraint in time slot t
T_n^t	Execution time of task n in time slot t
b_n^t	Battery level of MD$_n$ in time slot t
e_n^t	Harvested energy of MD$_n$ in time slot t
E_n^t	Energy consumption of task n in time slot t

4) latency constraint: Non-divisible and latency-sensitive tasks must be completed within a strict deadline to ensure QoE. Thus, we have:

$$T_n^{t} < \tau_n^{t}, \quad \forall n \in \mathcal{N}, t \in \mathcal{T}. \tag{11}$$

Here, T_n^t and τ_n^t denote the completion time and the latency constraint of task n in time slot t, respectively.

Remark: The constraints outlined in (1)–(9), (11) collectively establish a robust framework for task offloading and resource allocation in a hybrid-energy MEC system. They highlight the critical balance between computational efficiency and resource sustainability, enabling each MD to make informed offloading decisions based on its local battery status, latency requirements, and the resource availability of the MECSs and the cloud server. Furthermore, constraint (10) ensures that the expected battery level remains manageable over the long term, thereby preventing inefficiencies associated with energy overaccumulation. This ultimately guarantees the stability of the MEC energy management system.

B. Computation and Communication Model

By the offloading decisions listed in (1)–(3), each task must be performed either locally on an MD or remotely on an edge server or on a cloud server.

1) Task local execution: For local computing, let the computation capacitity of MD$_n$ be denoted as f_n'. In time slot t, MD$_n$ allocates a proportion $\alpha_n^t \in [0, 1]$ of its computing resources through dynamic voltage and frequency scaling. Thus, the local computing resource allocation can be expressed as $f_n^t = f_n' \alpha_n^t$. The task local execution latency for the task H_n^t on MD$_n$ in time slot t is given by:

$$T_n^{comp,t} = \frac{(1 - a_{nm}^{t}) c_n^t s_n^t}{f_n^t}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}', t \in \mathcal{T}. \tag{12}$$

The energy consumed to process task H_n^t on local MD_n is related to CPU frequency and is expressed as:

$$E_n^{comp,t} = \kappa_1(1 - a_{nm}^t)c_n^t s_n^t (f_n^t)^2, \forall n \in \mathcal{N}, m \in \mathcal{M}', t \in \mathcal{T}. \tag{13}$$

Here, κ_1 is the MD's energy consumption coefficient related to the chip architecture [33], and $\kappa_1(f_n^t)^2$ denotes the consumed energy per unit of computing resource.

2) Task offloading execution: For latency-sensitive tasks, effective task offloading heavily relies on the data transmission between MDs and MECSs. Here, MDs transmit on orthogonal channels. The wireless transmission from an MD to an MECS is affected by path loss and small-scale fading [34]. Once the task has been offloaded, it is sent to the selected edge server through a wired link.

The transmission latency relates to the allocated bandwidth and the transmission rate between MD_n and MECS_m in time slot t, denoted by $r_{nm}^{tran1,t}$ (in bits per second), which is defined by

$$r_{nm}^{tran1,t} = B_{nm}^{MECS} \log_2 \left(1 + \frac{P_{nm}^t \times g_{nm}^{MECS,t}}{\sigma^2}\right) \tag{14}$$

Here, B_{nm}^{MECS} is the bandwidth allocated for the transmission, P_{nm} is the transmit power, $g_{nm}^{\text{MECS},t}$ is the channel gain, which is a function of the distance between MD_n and MECS_m [28], and σ^2 represents the additive white Gaussian noise.

The transmission latency of task H_n^t is offloaded from MD_n to MECS_m in time slot t is given by:

$$T_n^{tran1,t} = \frac{s_n^t}{r_{nm}^{tran1,t}}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{15}$$

The consumed energy to offload task H_n^t from MD_n to MECS_m in time slot t is given by:

$$E_n^{tran1,t} = P_{nm}^t \times T_n^{tran1,t}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{16}$$

Similarly, we denote the transmission time and energy consumption from the m-th relay MECS_m to the cloud server in time slot t are $T_n^{tran2,t}$ and $E_n^{tran2,t}$, respectively. Then, we have

$$T_n^{tran2,t} = \frac{s_n^t}{r_{nm}^{tran2,t}}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{17}$$

and

$$E_n^{tran2,t} = P_m^t \times T_n^{tran2,t}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{18}$$

where $r_{nm}^{tran2,t}$ and P_m^t are transmission rate and transmission power from the m-th relay MECS_m to the cloud server.

Then, we denote $T_m^{comp1,t}$ and $T_m^{comp2,t}$ as the computation latency on MECS_m and the cloud server, respectively. $T_m^{comp1,t}$ and $T_m^{comp2,t}$ are defined by

$$T_n^{comp1,t} = \frac{(1 - x_{nm}^t)a_{nm}^t c_n^t s_n^t}{f_{nm}^t}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}, \tag{19}$$

and

$$T_n^{comp2,t} = \frac{x_{nm}^t a_{nm}^t c_n^t s_n^t}{f_m^t}, \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{20}$$

3) Total task computation overhead: For the sake of simplicity, as in [17], [28], and [36], energy consumption is primarily analyzed in terms of local computation and transmission, while other types are neglected. We focus on device energy because the devices are assumed to use green energy; thus, the consumed energy in our discussion represents the total energy for task completion, excluding that of edge/cloud servers. Notably, edge-to-cloud transmission energy is crucial as it impacts offloading choices and is therefore nonnegligible. For offloaded tasks, upon completing the computation, the selected server sends the result back to the MD. Given that the size of the calculation result is significantly smaller than the input data, the transmission time required to return the result is disregarded [17], [28], [35]. Finally, according to (12)-(20), the total consumed time T_n^t and consumed energy E_n^t of task n in time slot t are

$$T_n^t = \begin{cases} T_n^{\text{comp},t}, & \text{for local processing} \\ T_n^{\text{tran1},t} + T_n^{\text{comp1},t}, & \text{for edge processing} \\ T_n^{\text{tran1},t} + T_n^{\text{tran2},t} + T_n^{\text{comp2},t}, & \text{for cloud processing} \end{cases} \tag{21}$$

and

$$E_n^t = a_{n0}^t E_n^{comp,t} + \sum_{m=1}^{M} a_{nm}^t E_n^{tran1,t} + \sum_{m=1}^{M} x_{nm}^t E_n^{tran2,t} \tag{22}$$

If the MD's real-time latency demand exceeds a predefined deadline, a penalty term $\text{Pen}_{n,t}^d$ is applied to each MD, reflecting its contribution to the flexible demand. Similarly, when the battery level surpasses the defined threshold, the penalty term $\text{Pen}_{n,t}^e$ accounts for its contribution to the flexible adjustment. The penalty is formally defined as:

$$\text{Pen}_{n,t}^d = \begin{cases} -\alpha\omega_1[T_n^t - \tau_n^t]^+, & \forall n \in \mathcal{N}, \text{ if } T_n^t > \tau_n^t, \\ 0, & \forall n \in \mathcal{N}, \text{ otherwise.} \end{cases} \tag{23}$$

and

$$\text{Pen}_{n,t}^e = \begin{cases} -\beta\omega_2[b^{min} - b_n^t]^+, & \forall n \in \mathcal{N}, \text{if } b_n^t < b^{min}, \\ 0, & \forall n \in \mathcal{N}, \text{ otherwise.} \end{cases} \tag{24}$$

Here, the operators $[\cdot]^{+/-} = \max / \min\{\cdot, 0\}$ represent selecting the greater or lesser value between $\cdot$ and 0, respectively. ω_1 and ω_2 are the weighting factor.

C. Energy Harvesting Model

In this work, we investigate hybrid-energy MEC systems where MDs are supported by renewable green energy sources such as solar and wind. Each MD is equipped with an energy harvester to collect renewable energy [15]–[17], [21], [36], [49], utilizing stored energy for local processing and offloading tasks to MEC servers.

At the start of every time slot [15], we treat the energy harvesting (EH) process as receiving sequential energy packets . The harvested energy e_n^t is saved in a battery, and its arrival is intermittent and stochastic, as in [17], [36], and [49], typically modeled as a stochastic process. To simplify, we assume that

the energy harvested by each MD follows an independent and identically distributed process, with $e^t_{\max}$ serving as its upper limit. Specifically, e^t_n follows a continuous uniform distribution over the interval $[0, e^t_{\max}]$, where: - 0 aligns with physical reality (no energy harvesting); - $e^t_{\max}$ denotes the maximum harvestable energy per time slot, constrained by the MD's energy harvester hardware capacity (e.g., solar panel efficiency).

The battery status in the next time slot is determined by both energy consumption and harvesting. It evolves by

$$\widehat{b}^t_n = \min\left\{b^{max}_n, b^t_n + e^t_n\right\} \tag{25}$$

This equation updates the estimated battery level $\widehat{b}^t_n$ by incorporating the harvested energy e^t_n into the current battery level b^t_n. It ensures that the battery does not exceed its maximum capacity b^{max}_n.

To capture the dynamics of energy consumption, the battery status for the next time slot is modeled as follows:

$$b^{t+1}_n = \max\left\{0, \widehat{b}^t_n - E^t_n\right\} \tag{26}$$

Here, b^{t+1}_n represents the battery level of MD_n in time slot $t+1$ and E^t_n indicates the energy consumption for local computation and transmission of MD_n in time slot t. This iterative process ensures that the battery level accurately reflects both energy inflow and outflow dynamics within the hybrid-energy MEC system.

D. Problem Formulation

Based on above-mentioned models, the latency-aware resource-constrained scheduling problem can be described as follows:Given an MEC system with M MECSs, a cloud server and N MDs equipped with IoT components that generate real-time tasks, we aim to make optimal offloading decisions and allocate resources efficiently. These decisions involve determining whether a task should be performed locally on the MD, offloaded to one of the MECSs, or processed by the cloud. The objective is to minimize overall latency and consumed energy while satisfying long-term MEC energy constraints, thereby enhancing the QoE. The (LARCS) problem is mathematically rewritten as :

$$\begin{aligned} \textbf{(P1)} \min_{a^t_{nm}, f^t_{nm}} \quad & \lim_{T\to+\infty} \sum_{t=0}^{T-1} \sum_{n=1}^{N} \left(\alpha T^t_n + \beta E^t_n\right) \\ \text{s.t.} \quad & (1) - (11) \end{aligned} \tag{27}$$

where α and β are weighted parameters to make a trade-off between the latency and energy. It is obvious that **P1** is an MINLP model, as the objective function is non-linear and it involves both discrete decision variables a^t_{nm} and continuous decision variables f^t_{nm}. Solving **P1** is challenging due to the following reasons:

1) The unpredictable and dynamic nature of MDs' workloads and computing demands, coupled with stochastic parameters, complicates offloading scheduling in hybrid-energy MEC systems.
2) The discrete solution space is vast with N MDs generating tasks and M available MECSs, each task has $M+1$ choices, leading to an exponential action space of $O(N^{M+1})$.
3) MDs may have different preferences regarding latency and consumed energy. For instance, in virtual reality applications, processing latencys can have severe consequences, whereas energy efficiency is crucial in industrial and unmanned aerial networks.
4) Task offloading and resource allocation decisions are interdependent, governed by a non-convex, non-separable objective function and a continuous-discrete hybrid action space.

Theorem. LARCS is NP-completeness.

Proof: The 0–1 knapsack problem is NP-completeness [38]. LARCS is structurally identical to KNP, thus establishing a polynomial-time reduction, KNP $\leq_p$ LARCS. Since KNP is NP-completeness, all problems in NP can be reduced to LARCS, i.e., A $\in$ NP $\Rightarrow$ A $\leq_p$ LARCS. Additionally, LARCS is verifiable in polynomial time, confirming that LARCS is NP-completeness. ■

E. Problem Reformulation

We employ a complementary model-free method to better handle highly dynamic and diverse environments while maintaining long-term MEC efficiency. To fill the gap mentioned above, we design a scalable and model-free resource scheduling algorithm, leveraging EH techniques to enhance adaptability and scalability in long-term MEC optimization. Specifically, we transform **P1** into a Markov game [39], a multi-agent version of the Markov decision process [40], and develop a model-free, latency-aware, resource-constrained scheduling algorithm based on MADRL to solve this game, ensuring robust performance under varying conditions.

Since the MADRL approach discussed in Section IV-A and Section IV-B does not require knowledge of the state transition function, the three parts of the Markov game associated with **P1** are the state, action, and reward functions.

1) State: The observations of an agent characterize the state information of the environment, providing the necessary data for the agent's decision-making process. The observation $o_{n,t}$ for each agent n at time slot t is defined as $o_{n,t} = [H^t_n, e^t_n, b^t_n, g^t_n, P^t_n]$, where $H^t_n = (s^t_n, c^t_n, \tau^t_n)$, renewable energy harvesting e^t_n, battery status b^t_n, channel gain $g^t_n = (g^{MECS,t}_{nm}, g^{Cloud,t}_m), m \in \mathcal{M}$, transmit power $P_n = (P^t_{nm}, P^t_m), m \in \mathcal{M}$. This observation constitutes a subset of the global state of the hybrid-energy MEC environment s_t; that is, agent n can only observe partial information from the environment, as it cannot physically access the other agents' operating parameters. Consequently, the global state of the hybrid-energy MEC environment s_t is derived as the concatenation of all agents' observations at period t, expressed as $s_t = (o_{1,t}, o_{2,t}, \ldots, o_{N,t})$.

2) Action: The agent makes decisions according to the current state to interact with the environment. This study considers both task offloading decisions and overall resource allocation, leading to a mixed-action space. In this context, task offloading decisions are represented as discrete

actions, while resource allocation involves continuous actions. Thus, the action of all agents can be expressed as $a_t = (a_{1m}^t, a_{2m}^t, \ldots, a_{Nm}^t, f_{1m}^t, f_{2m}^t, \ldots, f_{Nm}^t)$. With the offloading decisions made by each MD, the system executes tasks in parallel, and each agent receives rewards from the environment based on the outcomes, transferring to the next state.

3) Reward: The basic reward $r_{n,t}^{latency}$ and $r_{n,t}^{energy}$ for agent n in time slot t is to minimize overall latency and energy consumption:

$$r_{n,t}^{latency} = -\alpha\omega_1 T_n^t \tag{28}$$

$$r_{n,t}^{energy} = -\beta\omega_2 E_n^t \tag{29}$$

Here, since latency and energy consumption are measured in different units, ω_1 and ω_2 serve as weighting factors to normalize the rewards of these physical quantities to a common range.

Furthermore, as demonstrated in [13], model-based system-centric coordination approaches ensure long-term benefits by enforcing explicit global constraints (corresponding to constraint (8)). However, in the analyzed model-free MADRL framework, constraints that couple multiple action dimensions cannot be explicitly enforced [41]. To address this issue, we develop an innovative reward-shaping approach that indirectly ensures compliance with long-term MEC energy constraints. Specifically, we introduce a penalty term to discourage abrupt fluctuations in energy consumption or computational load. These fluctuations, referred to as rebound peaks, occur when multiple MDs simultaneously trigger high-power computation or data transmission requests, leading to short-term resource congestion or excessive energy depletion. Such peaks can overload edge servers, degrade service quality, or cause energy deficits in hybrid-powered MEC systems. Our proposed penalty mechanism incorporates a penalty component $r_{n,t}^{pen}$ ($r_{n,t}^{pen} = \mathrm{Pen}_{n,t}^d + \mathrm{Pen}_{n,t}^e$) into each agent's reward function. Here, $\mathrm{Pen}_{n,t}^d$ accounts for the latency-induced peak effects as in (23), while $\mathrm{Pen}_{n,t}^e$ penalizes excessive energy bursts as in (24). By integrating this mechanism, we guide the learning process toward smooth and balanced resource allocation strategies, thereby preventing severe fluctuations that could compromise the efficiency and stability of the MEC system. As far as we are aware, this study is the first to explicitly incorporate long-term MEC energy constraints and rebound peak effects into a model-free RL framework, which has not been addressed in existing RL literature [15]-[22], [28], [31].

Finally, the final normalized reward function $\mathcal{R}_{n,t}$ of each agent n at t is:

$$\mathcal{R}_{n,t} = r_{n,t}^{latency} + r_{n,t}^{pen} + r_{n,t}^{energy} \tag{30}$$

Here, $r_{n,t}^{\text{latency}}$ and $r_{n,t}^{\text{energy}}$ convert the objective of minimizing latency and energy consumption in (27) into maximizing rewards via negative mapping. Meanwhile, $r_{n,t}^{\text{pen}}$ incorporates penalty terms for violations of the constraints in (8) and (11), i.e., it penalizes invalid solutions (e.g., task latency exceeding the deadline or battery level falling below the given threshold) and indirectly helps the algorithm pursue long-term benefits.

4) Problem Reformulation: The decision-maker's policy defines a mapping from states to actions, represented as $\pi : \mathcal{S} \to \mathcal{A}$. The reinforcement learning aims to determine an optimal policy to maximize the expected cumulative reward over time, defined as:

$$\textbf{(P2)} \quad \pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t\in\mathcal{T}}\sum_{n\in\mathcal{N}} \gamma^t \mathcal{R}_{n,t} \mid \pi\right], \quad \text{s.t.} \quad (1) - (11) \tag{31}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operator with respect to time-varying system parameters, such as computational demands and available resources.

IV. MADRL-BASED RESOURCE SCHEDULING ALGORITHM

To tackle the aforementioned Markov game, we employ MADRL to design a latency-aware resource-constrained scheduling algorithm with an attention mechanism. Specifically, we adopt multi-actor-attention-critic (MAAC) approach [42] for MADRL.

A. MAAC Approach

As part of the CTDE MADRL paradigm, MAAC differs from MADDPG by facilitating joint learning of critics through a shared set of learnable parameters, rather than training critics independently with private agent information. Furthermore, a multi-head attention module [44] from the Transformer is applied, allowing every agent to retrieve other agents for essential information on their observations and actions, which is then incorporated into the estimate of its value function. This design effectively addresses dimensionality issues arising from coordinated optimization among multiple users in MEC systems, offering better generalizability [18], [20]-[22], improved scalability [18], [20]-[22], reduced computational complexity [22], and enhanced user privacy preservation [12]-[32] than previous approaches.

1) SAC Method: The MAAC method is a multi-agent version of the SAC method, readers are encouraged to refer to [43] for details.

SAC is classified as an actor-critic single-agent deep reinforcement learning (SADRL) algorithm [41]. It utilizes a parameterized critic network Q^ψ that receives a state s and an action a as inputs, producing a scalar estimate of the Q-value function $Q^\psi(s,a)$. Simultaneously, the parameterized actor network π^ϕ takes the state s as input and generates the action selection probabilities $\pi^\phi(a|s)$.

To optimize the performance function $J(\pi^\phi)$, the policy parameters ϕ are updated by following the performance gradient $\nabla_\phi J\left(\pi^\phi\right)$, which is expressed as

$$\nabla_\phi J\left(\pi^\phi\right) = \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi}\left[\nabla_\phi \log\left(\pi^\phi\left(a \mid s\right)\right) Q^\psi\left(s,a\right)\right], \tag{32}$$

where $\mathcal{D}$ represents the experience replay buffer that stores experiences transitions $(s,a,r,\tilde{s})$. For each training iteration, a subset of experiences is randomly selected from this buffer, which contains the experiences of all agents.

The critic network performs policy evaluation by estimating the Q-value function. This is accomplished using temporal difference (TD) learning, minimizing the following mean-squared TD error:

$$L(\psi) = \mathbb{E}_{(s,a,r,\tilde{s})\sim\mathcal{D}}\left[\left(Q^{\psi}(s,a) - y\right)^2\right], \tag{33}$$

where $y = r(s,a) + \gamma\mathbb{E}_{\tilde{a}\sim\pi(\tilde{s})}[Q^{\bar{\psi}}(\tilde{s},\tilde{a})]$ and $Q^{\bar{\psi}}$ is the parameterized target Q-value function.

To enhance exploration and prevent premature convergence to non-optimal deterministic policies (as seen in [9]), the performance gradient is adjusted to incorporate an entropy term which also enhances both robustness and exploration capabilities, as shown in the following equation:

$$\nabla_{\phi} J\left(\pi^{\phi}\right) = \mathbb{E}_{s\sim\mathcal{D},a\sim\pi}\left[\nabla_{\phi}\log\left(\pi^{\phi}\left(a \mid s\right)\right)\rho\left(s,a\right)\right], \tag{34}$$

where $\rho\left(s,a\right) = -\beta\log\left(\pi^{\phi}\left(a \mid s\right)\right) + Q_n^{\psi}(o,a) - z\left(s\right)$, β is the temperature parameter that balances entropy maximization and reward, while $z\left(s\right)$ serves as a state-dependent baseline, highlighting the most beneficial agents for the overall task and thereby enhancing the learning policies for $Q^{\psi}(s,a)$. The target value y utilized in the loss function (33) can be rewritten as

$$y = r(s,a) + \gamma\mathbb{E}_{\tilde{a}\sim\pi(\tilde{s})}\left[Q^{\bar{\psi}}(\tilde{s},\tilde{a}) - \beta\log\left(\pi^{\tilde{\phi}}\left(\tilde{a} \mid \tilde{s}\right)\right)\right] \tag{35}$$

where $\pi^{\tilde{\phi}}\left(\tilde{a} \mid \tilde{s}\right)$ denotes the target policy function parameterized by $\tilde{\phi}$.

2) Principle of MAAC: The MAAC method extends the SAC algorithm to a multi-agent environment by adhering to the CTDE paradigm. This approach enables centralized training of the critics while allowing each agent to execute its learned policy independently. A key component of MAAC is an attention mechanism that lets each agent selectively integrate information from other agents' observations and actions into its own Q-value estimate. Specifically, as illustrated in Fig. 3, the Q-value $Q_n^{\psi}(o,a)$ for agent n is computed by factoring in contributions from other agents.

The action-value function $Q_n^{\psi}(o,a)$ relies only on agent n's local observation o_n, action a_n, and a contribution x_n derived from the abstract, learned representation of other agents' local situations. By using x_n in the Q-value function, the individual agent can make decisions informed by the collective behavior of other agents, yet without access to their specific local information, thereby preserving privacy. This structure enables agent n to incorporate collective insights while addressing privacy concerns, as x_n is a generalized, learned characterization rather than explicit private data from other agents. $Q_n^{\psi}(o,a)$ for agent n is calculated by:

$$Q_n^{\psi}(o,a) = f_n(g_n(o_n,a_n),x_n) \tag{36}$$

where f_n is a two-layer multi-layer perceptron (MLP) and $g_n(o_n,a_n) = e_n$ is a one-layer embedding MLP, both parameterized specifically for each agent. Importantly, f_n and g_n contain learnable parameters unique to each agent, which are private and inaccessible to others.

The contribution, integrating state-action information from all other agents to agent n, is computed by:

$$x_n = \sum_{j\in n-}\omega_j v_j = \sum_{j\in n-}\omega_j h(W_{\nu}e_j), \tag{37}$$

where $n-$ represents the index set of all agents other than n. W_{ν} is a common matrix for all agents, transforming an agent j's embedding e_j into a value, and $h(\cdot)$ is a nonlinear activation function. The attention weight ω_j attached to agent j's value (indicating the degree of attention agent n pays to agent j's value) is obtained by comparing the similarity between agent n's embedding e_n and other agents' embeddings e_j, then applying a softmax operation:

$$\omega_j = \frac{\exp((W_k e_j)^T W_q e_n)}{\sum_{j=1}^{N}\exp((W_k e_j)^T W_q e_n)}, \tag{38}$$

where W_q and W_k are shared, learnable matrices transforming e_n/e_j into a query/key, respectively. Every agent is equipped with a multi-head self-attention mechanism where every attention head is configured with the parameters (W_q, W_k, W_{ν}) to generate aggregated contributions from other agents to agent n. Contributions from all attention heads are then combined to form a vector $x_i = (x_1,\ldots,x_n)$. By using different parameters for each attention head, it is possible to focus on information from other agents from diverse perspectives.

Since three learnable parameters (W_q, W_k, W_{ν}) are shared, we update all critics to minimize a joint regression loss function as below.

$$L(\psi) = \sum_{n=1}^{N}\mathbb{E}_{(o,a,r,\tilde{o})\sim\mathcal{D}}\left[\left(Q_n^{\psi}(o,a) - y_n\right)^2\right], \tag{39}$$

where

$$y_n = r_n(o,a) + \gamma\,\mathbb{E}_{\tilde{a}\sim\pi^{\bar{\phi}}(\tilde{o})}\left[Q_n^{\bar{\psi}}(\tilde{o},\tilde{a}) - \beta\log\left(\pi^{\bar{\phi}_n}(\tilde{a}_n \mid \tilde{o}_n)\right)\right].$$

With shared learnable parameters, gradients from the experiences of all agents contribute to updates to these specific attention weights during joint backpropagation, as shown in Fig. 3.

Similarly, for the actor networks, every agent updates the weights π^{ϕ_n} of its own network. The gradient for updating individual policies is

$$\nabla_{\phi_n} J\left(\pi^{\phi_n}\right) = \mathbb{E}_{(o,a,r,\tilde{o})\sim\mathcal{D}}\left[\nabla_{\phi_n}\log\left(\pi^{\phi_n}\left(a_n \mid o_n\right)\right)\rho_n\left(o_n,a_n\right)\right], \tag{40}$$

where

$$\rho_n\left(o_n,a_n\right) = -\beta\log\left(\pi^{\phi_n}\left(a_n \mid o_n\right)\right) + Q_n^{\psi}(o,a) - z\left(o,a_{n-}\right),$$

with

$$z\left(o,a_{n-}\right) = \mathbb{E}_{a_n\sim\pi^{\phi_n}(o_n)}\left[Q_n^{\psi}\left(o,(a_n,a_{n-})\right)\right].$$

The term $Q_n^{\psi}(o,a) - z\left(o,a_{n-}\right)$ represents the multi-agent advantage function, a multi-agent baseline that highlights the more helpful agents for the overall task, thereby promoting a more effective learning policy. This function compares the value of a specific action a_n to the mean value among all actions of agent n (with other agents' actions fixed), indicating whether the current action a_n enhances the expected return.

Both the actor and target critic networks are updated with a soft update rule to periodically align their parameters with the main networks. This update is defined as:

$$\begin{aligned} \bar{\psi} &= \tau\bar{\psi} + (1-\tau)\psi \\ \bar{\phi}_n &= \tau\bar{\phi}_n + (1-\tau)\phi_n \end{aligned} \tag{41}$$

where $\tau \in (0, 1]$ is the soft update coefficient, controlling the rate of parameter adjustment.

Finally, each agent in the LARC-MADRL algorithm continuously collects data and minimizes loss function (39) to converge to an optimal solution. By integrating the above updates, the final LARC-MADRL algorithm is obtained.

B. The Proposed LARC-MADRL Algorithm

Algorithm 1 Training Phase of LARC-MADRL

Input: MEC system parameters
Output: Optimized actor parameters ϕ, critic parameters ψ

1: Set the environments with N agents;
2: Set the size of $\mathcal{D}$;
3: Set target networks $Q_n^{\bar{\psi}}$ and $\pi^{\bar{\phi}_n}$ by: $\bar{\psi} \leftarrow \psi, \bar{\phi} \leftarrow \phi$;
4: **for** episode = 1 to P **do**
5: Reset environments, and get initial observation state $o_{n,0}$ for every agent n;
6: **for** $t = 0$ to $T-1$ **do**
7: Select actions $a_{n,t} \sim \pi^{\phi_n}(\cdot \mid o_{n,t})$ for $n \in \mathcal{N}$;
8: Transmit actions $a_{n,t}$ to all parallel environments and receive $o_{n,t+1}$;
9: Calculate basic rewards $r_{n,t}^{latency}$ and $r_{n,t}^{energy}$ by (28) and (29);
10: Calculate penalty terms $\mathrm{Pen}_{n,t}^{d}$ and $\mathrm{Pen}_{n,t}^{e}$ by (23) and (24);
11: Calculate normalized reward $\mathcal{R}_{n,t}$ by (30);
12: Save (o_t, a_t, r_t, o_{t+1}) in $\mathcal{D}$;
13: **if** $B_{\text{memory}} \geq B_{\text{size}}$ and $t\%T_{\text{update}} = 0$ **then**
14: Sample mini-batch $\mathbf{B}$ of size B_{size} from $\mathcal{D}$;
15: Compute $Q_n^{\psi}(o_n^l, a_n^l)$ for all n in parallel, $1 \leq l \leq B_{\text{size}}$, where a_n^l and o_n^l denote the l-th a_n and o_n in $\mathbf{B}$;
16: Calculate $\tilde{a}_{n_-}^l \sim \pi^{\bar{\phi}_n}(\tilde{o}_n^l)$ for all n and l;
17: Calculate $Q_n^{\bar{\psi}}(\tilde{o}_n^l, \tilde{a}_n^l)$ for all n and l;
18: Reset critic by minimizing loss function (39);
19: Set $a_n^l \sim \pi^{\bar{\phi}_n}(o_n^l)$ for all n in parallel;
20: Set $Q_n^{\psi}(o_n^l, a_n^l)$ for all l and n;
21: Reset actor via policy gradient (40);
22: Reset target networks using (41);
23: **end if**
24: **end for**
25: **end for**

The LARC-MADRL algorithm includes two sub algorithms: the training algorithm (Algorithm 1) and the execution algorithm (Algorithm 2). In Algorithm 1, agent n interacts with the hybrid-energy MEC environment at each time step t. The transition tuple $(o_{n,t}, a_{n,t}, r_{n,t}, o_{n,t+1})$ is collected and stored in the experience replay buffer $\mathcal{D}$ (lines 7-12). When the buffer size B_{memory} exceeds the batch size B_{size}, a mini-batch is randomly sampled every T_{update} slots to train the critic and actor networks. This technique, known as experience replay, offers several benefits, including improved data efficiency, reduced update variance, and increased stability. Lines 15-18 of Algorithm 1 describe the update of the critic network, while lines 19-21 cover the actor network update. Finally, the target network weights are updated in line 22. All actors share the identical network, which comprises an input layer, multiple hidden layers with Leaky ReLU activation functions, and an output layer with a softmax activation function. Similarly, all critics follow the identical architecture.

To address non-stationary environments, the algorithm dynamically resets the random exploration probability to one when environmental changes are detected, allowing agents to resume effective exploration and adapt their policies.

Fig. 2 presents the proposed LARC-MADRL method.

Once the training is complete, the actor network weights remain fixed, and the learned policies are applied during actual testing, as described in Algorithm 2. During testing, every actor selects $a_{n,t}$ based on its current local observation $o_{n,t}$, and the MEC system executes these actions and gain reward. After each time step, the next local observation $o_{n,t+1}$ is obtained, and this process continues until the testing phase concludes. Throughout the execution, there is no communication among the agents. Instead, energy management and real-time decision-making are coordinated in a fully decentralized manner, with every agent independently making decisions through its actor network. Since decisions are based solely on the current observation o_t, the proposed algorithm requires no prior knowledge of uncertain parameters and operates independently of hybrid-energy dynamic models. Note that the proposed learning architecture is applicable to both the system described in Section III-A and other similar hybrid decision-based models.

Algorithm 2 Execution Phase of LARC-MADRL

Input: ϕ *// The actor network weights*
Output: Action a_t

1: $o_1 \leftarrow (o_{1,1}, \ldots, o_{N,1})$ *// Each agent receives initial local observation (task status, battery and energy status, channel gain, and transmit power)*
2: **for** $t = 0$ to H_{test} **do**
3: **for** each agent n in parallel **do**
4: Select action $a_{n,t}$ using the learned policy $\pi^{\phi_n}(\cdot \mid o_{n,t})$ at the start of slot t;
5: Take action $a_{n,t}$, affecting the MEC system;
6: Calculate basic rewards $r_{n,t}^{latency}$ and $r_{n,t}^{energy}$ by (28) and (29);
7: Calculate penalty terms $\mathrm{Pen}_{n,t}^{d}$ and $\mathrm{Pen}_{n,t}^{e}$ by (23) and (24);
8: Calculate normalized reward $\mathcal{R}_{n,t}$ by (30);
9: Receive new observation $o_{n,t+1}$ when time slot t ends;
10: **end for**
11: **end for**

For practical deployment, Algorithm 1 (Training Phase)

is executed offline on a centralized cloud/edge server with sufficient computing resources. First, the MEC system parameters (e.g., number of MDs/MECSs, computation capacities, energy harvesting models) are initialized based on real-world deployment (e.g., IoT devices with solar harvesters, base stations integrated with MEC servers). Historical data (task arrival patterns, energy harvesting, channel gains) is collected to pre-train the actor-critic networks, with experience replay buffers stored in distributed storage (e.g., edge server) for efficient sampling. The multi-head attention module's shared parameters are synchronized across agents via secure communication protocols to preserve privacy, as agents only exchange embeddings not raw battery/task data. Algorithm 2 (Execution Phase) runs in real time on MDs and MEC servers in a decentralized manner. Each MD's actor network is deployed as a lightweight inference model (optimized via model pruning/quantization for resource-constrained devices) that processes local observations (task status, battery and energy status, channel gain, transmit power) to output offloading decisions. MEC servers execute resource allocation based on received actions, with task execution prioritized via the reward-shaped penalty mechanism to mitigate energy rebound peaks and surpass task deadlines.

It is constructive to clarify that in the hybrid-energy MEC system, the environment is viewed as a black box by the MD agents interacting with it. This abstraction enables agents to engage with the MEC system without instead of understanding its internal mechanisms, assuming it functions as a trustworthy third party that manages resources and services. To safeguard against potential security threats, such as data interception or spoofing, a secure communication channel between agents is required, with all transmitted information encrypted within the MEC system [45]. Furthermore, to protect the privacy of MDs, the proposed framework utilizes information embedding, which eliminates the need for agents to directly or repeatedly exchange sensitive data, thus reducing the risk of data exposure and improving overall system efficiency.

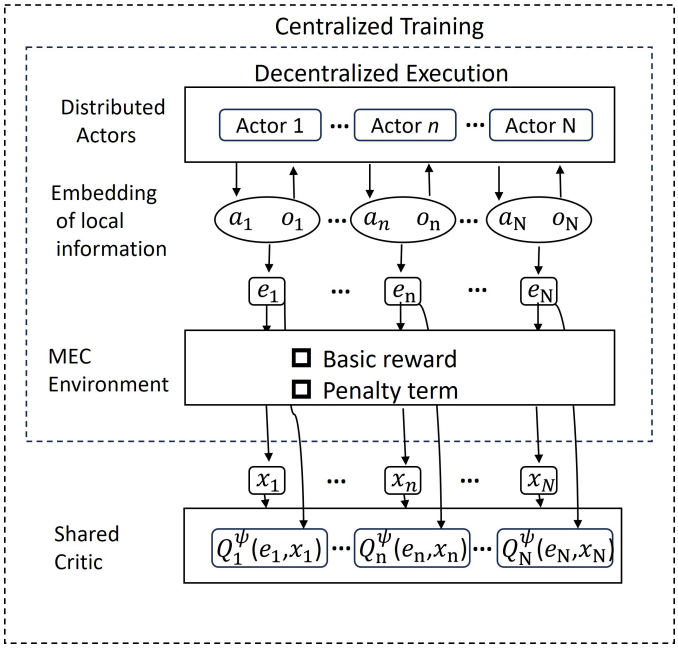

Fig. 2. The CTDE architecture of LARC-MADRL

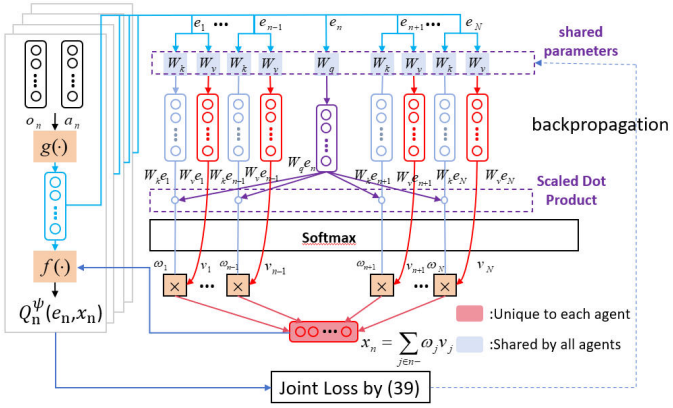

Fig. 3. Attention-based Q-value function estimate under LARC-MADRL

V. PERFORMANCE EVALUATION

TABLE III
KEY FEATURES AND ADVANCEMENTS IN COMPARATIVE ALGORITHMS

Feature	Algorithm				
	RS	DQN	MPO	MADDPG	Proposed
Non-Stationary	No	Yes	No	No	Yes
Generalization Capability	No	Yes	Yes	Yes	Yes
Privacy Preserving	No	No	No	No	Yes
Adaptability	No	Yes	Yes	Yes	Yes
Robustness	No	Yes	Yes	Yes	Yes
Scalability	No	Yes	Yes	Yes	Yes

A. Experimental Setup

We have made numerical experiments on a computer with an AMD R5 5600H processor at 3.30 GHz, 16 GB RAM, and Windows 11. In the experiment, M = 3 MECSs are deployed to provide edge computing capabilities for MDs. The size of task data is randomly selected from 1 MB to 50 MB. One task per time slot issues from every MD, which can be processed locally with a computational frequency f_n^t ranging from 0.4 GHz to 1.5 GHz, offloaded to a mobile edge computing server with a fixed frequency f_{nm}^t of 2 GHz, or offloaded to a cloud server with a fixed frequency f_m^t of 6 GHz. The channel noise power σ^2 was set to 5×10^{-27}, the maximum harvested energy was set to $e_{\max}^t = 1$ mJ and the battery thresholds were defined as $b^{\max} = 3.2$ mJ, and $b^{\min} = 0.5$ mJ. For simplicity, it is assumed that the wireless channel gain $g_{nm}^{\mathrm{MECS},t}$ follows a uniform distribution over the interval [5,14]$\cdot\sigma^2$ dB. Transmit power P_{nm}^t is in the range [1,24] dBm. The experimental data settings are derived from [16], [17], [25], [28], and [36]. The algorithm parameters included weight factors $w_1 = 25$ and $w_2 = 1$, a maximum deadline of MD$_n$ τ_n^t randomly chosen between 0.1s and 1s, a replay buffer size $\mathcal{D} = 10,000$, a

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. 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/TASE.2026.3690812

discount factor $\gamma = 0.98$, a maximum of $P = 500$ episodes with $T = 100$ time steps per episode, a soft target update factor $\tau = 0.0001$, and an update interval $T_{\text{update}} = 50$. Additionally, our attention-based critics incorporate a total of 4 attention heads in their architecture.

B. Benchmark Models

To fairly assess the performance of the proposed algorithm, we implement and compare it with four existing resource scheduling methods within the same hybrid-energy MEC environment. The key features and advancements of the comparative algorithms are listed in Table III. The comparative algorithms are as follows:

- Random scheme (RS). It addresses problem **P1** by randomly selecting offloading tasks while adhering to constraints of storage space and computing capacity to identify feasible solutions.
- DQN [18]. It employs Q-learning to determine the best actions in given states, utilizing deep or convolutional neural networks to estimate the Q-value function, effectively addressing high-dimensional state space challenges.
- Maximum power offloading (MPO) [46]. MDs offload their tasks using maximum transmission power. This strategy underscores the importance of optimizing power allocation in hybrid-energy MEC systems, providing a baseline for evaluating more sophisticated power management techniques.
- MADDPG [22]. It uses a CTDE architecture to to coordinate multiple agents during training while allowing MDs to make independent task offloading decisions in the execution phase. This approach enables agents to learn a shared policy while preserving individual decision-making capabilities, making it particularly effective in environments with complex interactions and varying objectives.

  Note: Complete ablation studies and analysis of extreme scenarios are included in the supplementary file due to space limits.

C. Convergence Performance of the Algorithm

Fig. 4 illustrates the convergence performance of various schemes with seven MDs, three MECSs, and one cloud server. Initially, the schemes have no environmental knowledge, and actions are almost randomly selected. Once enough samples are accumulated within a certain time, the schemes begin training the network. The results imply that our proposed scheme converges within approximately 200 episodes, maintaining a relatively stable curve, whereas the baseline algorithms continue to fluctuate. This stability is enabled by the attention mechanism and the multi-agent advantage function, which ensure consistency during policy updates [42]. Additionally, since the SAC method encourages exploration and the system parameters change for every episode whose reward varies in a narrow range. In contrast, the convergence curves of DQN and MADDPG are more oscillatory, with DQN learning more slowly and MADDPG struggling to converge due to complex

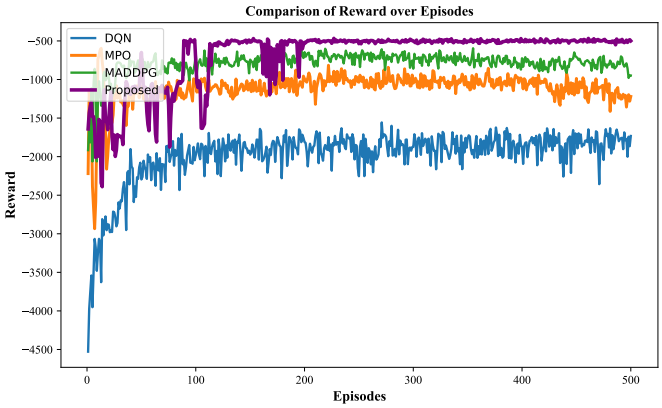

Fig. 4. Convergence Performance.

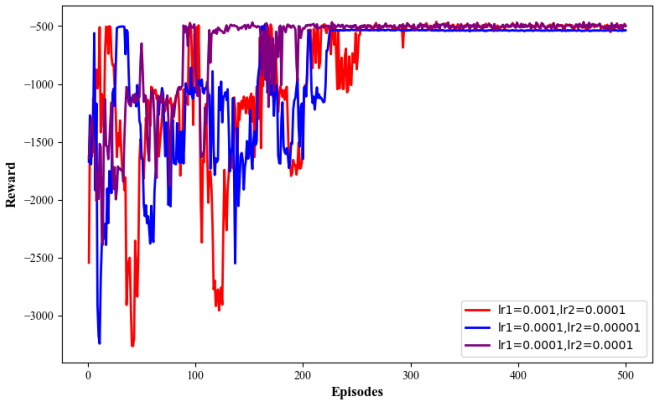

Fig. 5. The effect of different rates on the reward.

parameter tuning. In summary, the proposed algorithm demonstrates better convergence performance and quickly finds the optimal policy.

Fig. 5 gives the convergence of the proposed algorithm with various learning rates where $lr1$ is the actor network learning rate and $lr2$ is the critic network learning rate. The red line represents the case with $lr1 = 1 \times 10^{-3}$ and $lr2 = 1 \times 10^{-4}$, while the blue line corresponds to $lr1 = 1 \times 10^{-4}$ and $lr2 = 1 \times 10^{-5}$. The purple line shows that the count of of episodes has impact on the reward, achieving fast convergence with $lr1 = lr2 = 1 \times 10^{-4}$. The figure demonstrates that as the count of episodes increases, the reward of the proposed scheme stabilizes, and our algorithm effectively solves the complex problem **P1**.

D. Online Scheduling Performance of the Algorithm

1) Task penalty mechanism for guiding MEC energy consumption: long-term MEC energy constraints, rebound peaks, and real-time decision-making in MEC systems introduce complexity into resource allocation and task offloading. To meet this challenge, a novel reward-shaping mechanism (details in Section III-E) is proposed to leverage task penalty to regulate real-time energy usage for an MEC system effectively.

As seen in Fig. 6, when the MEC system experiences low energy consumption during a specific time slot t, the task penalty for that slot remains low or even zero, as observed in time slots 0, 13, 18, 24, and 41. In contrast, high energy consumption during time slot t triggers an increased task penalty, which subsequently leads to reduced energy usage in the following time slot $t+1$. For instance, substantial energy consumption in time slots 4, 10, 23, 32, and 42 results in elevated task penalties, effectively lowering energy usage in the subsequent time slots 5, 11, 24, 33, and 43. However, since the task penalty is not the sole influencing factor, high energy consumption in time slot t may result in only a partial decrease in the subsequent time slot $t+1$, where energy usage remains high but shows a downward trend. The adaptability of the task penalty ensures its continued influence, maintaining a high value to further guide energy reduction. For example, in time slot 3, energy consumption is high, resulting in a high task penalty. In the following time slot 4, energy consumption decreases but remains elevated, prompting the task penalty to stay high and guide further reductions in time slot 5. This real-time feedback mechanism, using the task penalty as an energy deficit indicator, ensures that the MEC system adheres to long-term MEC energy constraints effectively and dynamically.

2) Effect of parameters α and β on algorithmic performance: The generalizability of the proposed algorithm is validaed through its performance with different reward-weighted parameters. The algorithm's performance under parameters α and β is shown in Fig. 7. Given that $\alpha + \beta = 1$, as α increases and β decreases, the agent prioritizes latency reduction, which leads to higher energy comsumption. This behavior is expected, as α and β indicates the importance weight of latency and energy consumption, respectively. Thus, the algorithm offers a flexible trade-off between latency and energy consumption. In practice, α and β can be selected upon the specific requirements for latency and energy consumption, making LARC-MADRL suitable for both time-sensitive and energy-sensitive tasks. Furthermore, this indicates that the proposed algorithm can adapt in real time to different preferences regarding energy consumption and latency in dynamic environments.

E. Impact of a Variable Number of MDs on Algorithmic Performance

The scalability of the proposed algorithm is validated by its performance in terms of mean energy consumption, latency, and average dropout rate. In Figs. 8, 9, and 10, the performance of the RS algorithm consistently falls behind the other algorithms as the count of MDs grows, revealing the importance of a well-designed resource scheduling strategy and effective agent coordination in hybrid-energy systems. This underperformance occurs because the RS algorithm makes decisions without considering the current state or learning from past experiences. Unlike traditional algorithms, which rely on specific rules or heuristics, or RL algorithms, which optimize decisions over time based on feedback, the random scheme lacks strategy, adaptability, and the ability to learn, resulting in inefficient and suboptimal performance. Since the proposed algorithm effectively handles unknown dynamic models and agent coordination, it outperforms benchmark algorithms as the count of MDs increases. To be specific, when the count of MDs is 12, compared to the RS algorithm, the proposed scheme decreases the mean energy consumption, mean latency, and dropout rate by 77.23%, 37.59%, and 83.06%, respectively. Compared to DQN, these reductions are 43.07%, 36.88%, and 78.03%; compared to MPO, these reductions are 12.63%, 2.22%, and 4.80%; and compared to MADDPG, 28.72%, 6.72%, and 20.37%. The proposed algorithm obtains the best performance by fully utilizing the resources of MDs, edge servers, and the cloud server. These resources are effectively managed and flexibly allocated to different users according to their real requirements through the MADRL-based approach. Moreover, the intelligent algorithms enables the efficient resource management across MDs, edge servers, and the cloud server.

To evaluate computational cost and execution time fairly, we compare the average training time of the MADRL algorithm in the same environment (noting that the neural network inputs differ between SADRL and MADRL). As shown in Fig. 11, when the count of MDs grows, the average training time of MPO and MADDPG increases exponentially, while our proposed algorithm shows only a minor increase. This is because more users expand the solution space and solution complexity, leading to increased computational costs.

MPO and MADDPG critics process all information indiscriminately, whereas our approach enables joint learning of all agents' critics by using a common set of learnable parameters, thereby alleviating the exponential growth in coordination complexity. Furthermore, by utilizing an attention mechanism, our method selectively focuses on relevant agents and compresses the information into a fixed-size vector, resulting in improved scalability as the count of agents increases. Specifically, our method reduces the critic's input dimensions from $N\,|o_n + a_n|$ (the state-action information of all agents) to the connected dimensions e_n and x_n, effectively avoiding any dimensionality explosion in the state-action space. This reduces the critic's input dimension to $|e_n + x_n|$, enhancing scalability and reducing computational complexity by achieving a linear input space increase with regard to the count of agents, as opposed to the quadratic growth in existing MADRL approaches [15], [17], [22], [46]. Additionally, state and action inputs o and a may contain physical quantities that could reveal sensitive information about users, such as their preferences, habits, or locations. Compared to previous work [12]-[32], which lacks privacy protection measures in resource scheduling, our method enhances privacy protection by transforming o and a into e and x through an attention mechanism, effectively acting as an encryption method that safeguards user privacy during the scheduling process.

VI. CONCLUSION

Overall, we study the latency-aware resource-constrained scheduling problem in hybrid-energy MEC systems with non-divisible and latency-sensitive tasks, aiming to optimize latency and energy consumption while ensuring long-term

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. 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/TASE.2026.3690812

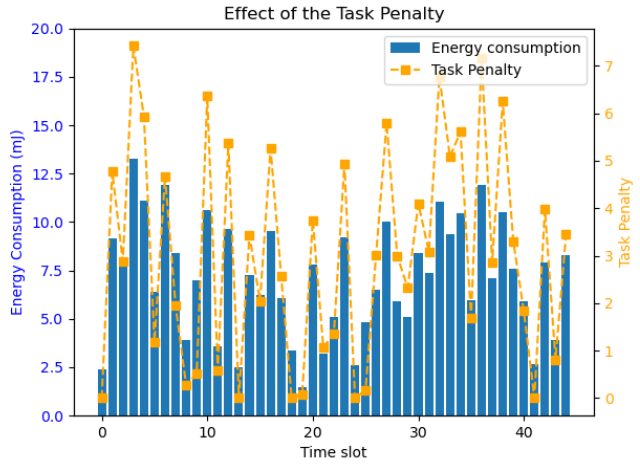

Fig. 6. Effect of the Task Penalty

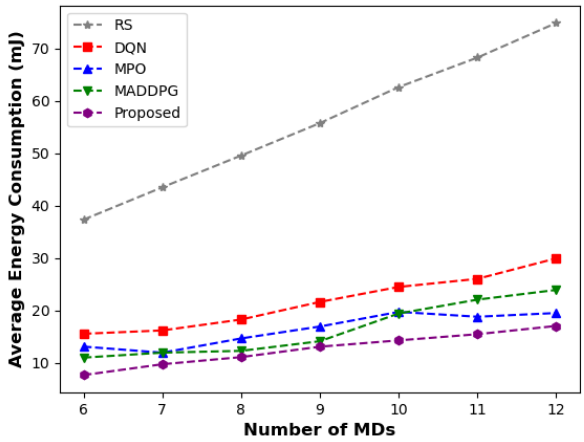

Fig. 8. Average Energy Consumption vs Number of MDs

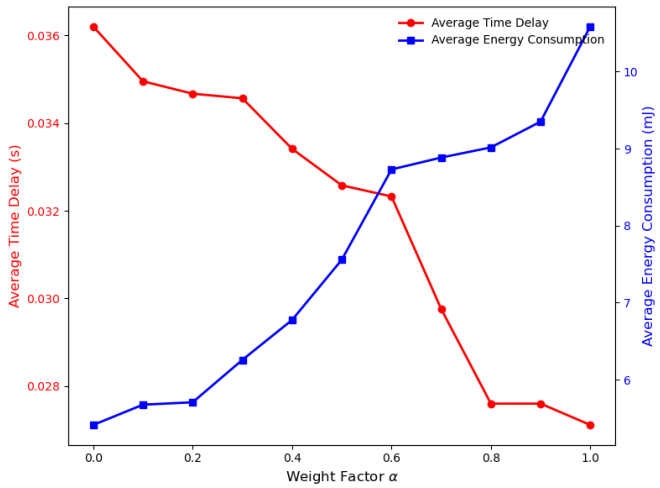

Fig. 7. Average Energy Consumption and Time latency vs Weight Factor α

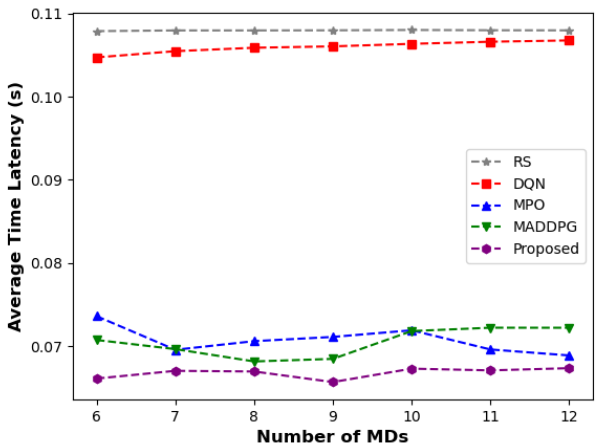

Fig. 9. Average Time latency vs Number of MDs

MEC energy constraints to enhance end-user QoE. Since the problem is an MINLP with tightly coupled optimization decision variables, conventional optimization methods struggle to efficiently find an optimal solution. To tackle this mathematically intractable challenge, we propose a scalable, model-free scheduling algorithm upon multi-agent DRL via an attention mechanism, enabling mobile devices to make decentralized and efficient offloading decisions. To ensure compliance with long-term MEC energy constraints, we design a novel reward shaping mechanism to penalize tasks and monitor energy deficits, effectively guiding real-time energy consumption. The attention mechanism plays a critical role in this framework by selectively focusing on relevant encoded data for critic estimation. Moreover, it facilitates joint critic learning across agents through shared trainable parameters, addressing challenges such as system dynamics, computational complexity, and privacy issues. Experimental results confirm the effectiveness of the proposed algorithm in significantly reducing latency and energy consumption while adhering to system constraints.

Despite these contributions, this work has several limitations that merit attention and future improvements. First, while the multi-head attention mechanism in LARC-MADRL reduces computational complexity compared to traditional MADRL methods, it still introduces non-negligible overhead during the training phase. For resource-constrained edge devices (e.g., low-cost IoT sensors with limited computing power and battery capacity), the inference speed may degrade in ultra-large-scale scenarios, as the attention module requires selective information aggregation across agents. Second, the reward-shaping mechanism relies on manually calibrated weighting factors ($\omega_1 = 25$, $\omega_2 = 1$) to normalize latency and energy metrics (Section III-E). The adaptability of these parameters to diverse application scenarios (e.g., varying task latency requirements, energy harvesting rates, or MEC server capacities) remains unvalidated, requiring further generalization. Third,

This article has been accepted for publication in IEEE Transactions on Automation Science and Engineering. 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/TASE.2026.3690812

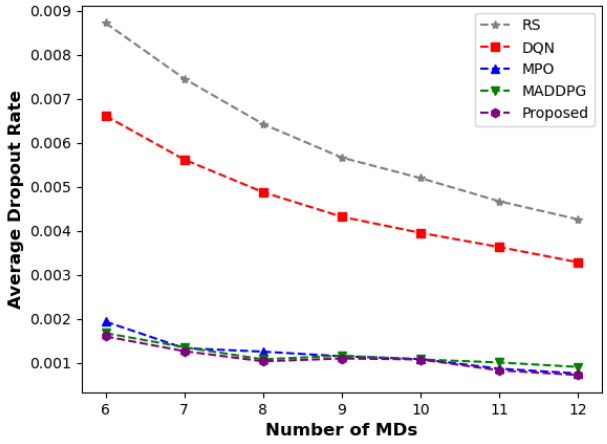

Fig. 10. Average Dropout Rate vs Number of MDs

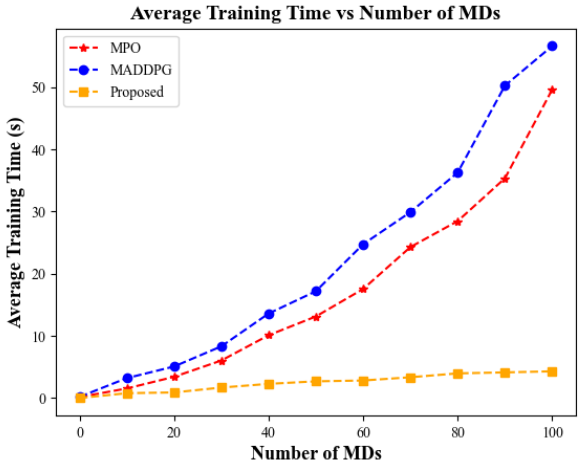

Fig. 11. Average Training Time vs Number of MDs

LARC-MADRL focuses on single-task scheduling per MD (one task per time slot per MD, Section III-A), but does not address the scheduling of multiple concurrent tasks per MD—a common scenario in practical IoT applications where devices generate heterogeneous workloads.

Building on the findings and limitations identified, future work will focus on three directions. First, we will explore lightweight attention variants (e.g., sparse attention, linear attention) to reduce computational overhead, enabling deployment on resource-constrained edge devices. Second, we will develop an adaptive reward-shaping mechanism with self-tuning parameters, leveraging meta-learning to optimize weight factors across diverse scenarios automatically. Third, we will extend the algorithm to support multi-task scheduling per MD and investigate integrated space-air-ground MEC networks for seamless global connectivity [47].

REFERENCES

[1] A. Abdo, T. S. Karamany, and A. Yakoub, "A hybrid approach to secure and compress data streams in cloud computing environment," *J. King Saud Univ.-Comput. Inf. Sci.*, pp. 101999, 2024.

[2] D. H. Nam, "A comparative study of mobile cloud computing, mobile edge computing, and mobile edge cloud computing," in *Proc. 2023 Congress in Computer Science, Computer Engineering, Applied Computing (CSCE)*, 2023, pp. 1219–1224.

[3] R. Cárdenas, P. Arroba, J. L. Risco-Martín, and J. M. Moya, "Modeling and simulation of smart grid-aware edge computing federations," *Cluster Comput.*, vol. 26, no. 1, pp. 719–743, 2023.

[4] N. Yang, J. Wen, M. Zhang, and M. Tang, "Multi-objective deep reinforcement learning for mobile edge computing," in *Proc. 2023 21st Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Networks (WiOpt)*, 2023, pp. 1–8.

[5] J. Mei, Z. Tong, K. Li, L. Zhang, and K. Li, "Energy-efficient heuristic computation offloading with delay constraints in mobile edge computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 4404–4417, 2023.

[6] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, W. Xu, K. Wei, and X. Jia, "Mobility-aware utility maximization in digital twin-enabled serverless edge computing," *IEEE Trans. Comput.*, 2024.

[7] S. Zhang, Q. Liu, K. Chen, B. Di, H. Zhang, W. Yang, D. Niyato, Z. Han, and H. V. Poor, "Large models for aerial edges: An edge-cloud model evolution and communication paradigm," *IEEE J. Sel. Areas Commun.*, 2024.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[9] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[10] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[11] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," *Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[12] A. Mohajer, M. S. Daliri, A. Mirzaei, A. Ziaeddini, M. Nabipour, and M. Bavaghar, "Heterogeneous computational resource allocation for NOMA: Toward green mobile edge-computing systems," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1225–1238, 2022.

[13] H. Jiang, X. Dai, Z. Xiao, and A. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4000–4015, 2023.

[14] J. Zhang, X. Zhou, T. Ge, X. Wang, and T. Hwang, "Joint task scheduling and containerizing for efficient edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 2086–2100, 2021.

[15] A. Daghayeghi and M. Nickray, "Decentralized computation offloading via multi-agent deep reinforcement learning for NOMA-assisted mobile edge computing with energy harvesting devices," *J. Syst. Archit.*, vol. 151, pp. 103139, 2024.

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

[17] T. Z. Gebrekidan, S. Stein, and T. J. Norman, "Combinatorial client-master multiagent deep reinforcement learning for task offloading in mobile edge computing," in *Proc. 23rd Int. Conf. Autonomous Agents Multiagent Syst. (AAMAS)*, 2024, pp. 2273–2275.

[18] C. Wang, D. Deng, L. Xu, and W. Wang, "Resource scheduling based on deep reinforcement learning in UAV assisted emergency communication networks," *IEEE Trans. Commun.*, vol. 70, no. 6, pp. 3834–3848, 2022.

[19] Y. Sun and Q. He, "Joint task offloading and resource allocation for multi-user and multi-server MEC networks: A deep reinforcement learning approach with multi-branch architecture," *Eng. Appl. Artif. Intell.*, vol. 126, pp. 106790, 2023.

[20] Y. Liu, J. Yan, and X. Zhao, "Deep reinforcement learning based latency minimization for mobile edge computing with virtualization in maritime UAV communication network," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4225–4236, 2022.

[21] T. Li, Y. Liu, T. Ouyang, H. Zhang, K. Yang, and X. Zhang, "Multi-hop task offloading and relay selection for IoT devices in mobile edge computing," *IEEE Trans. Mobile Comput.*, 2024.

[22] J. Du, Z. Kong, A. Sun, J. Kang, D. Niyato, X. Chu, and F. R. Yu, "MADDPG-based joint service placement and task offloading in MEC empowered air-ground integrated networks," *IEEE Internet Things J.*, 2023.

[23] W. Fan, L. Zhao, X. Liu, Y. Su, S. Li, F. Wu, and Y. Liu, "Collaborative service placement, task scheduling, and resource allocation for task offloading with edge-cloud cooperation," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 238–256, 2022.

[24] X. Ma, A. Zhou, S. Zhang, Q. Li, A. X. Liu, and S. Wang, "Dynamic task scheduling in cloud-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2116–2130, 2021.

[25] C. L. Chen, C. G. Brinton, and V. Aggarwal, "Latency minimization for mobile edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2233–2247, 2021.

[26] S. Tuli, S. R. Poojara, S. N. Srirama, G. Casale, and N. R. Jennings, "COSCO: Container orchestration using co-simulation and gradient based optimization for fog computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 101–116, 2021.

[27] H. Ma, P. Huang, Z. Zhou, X. Zhang, and X. Chen, "GreenEdge: Joint green energy scheduling and dynamic task offloading in multi-tier edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 4322–4335, 2022.

[28] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, and M. Xiao, "Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15513–15526, 2023.

[29] M. Z. Jiang, A. B. Lu, Q. H. Zhu, and L. K. Fei, "Delay-aware and energy-efficient task offloading based on adaptive large neighborhood search," in *Proc. 2023 IEEE Int. Conf. Syst., Man, Cybernetics (SMC)*, 2023, pp. 5015–5020.

[30] G. Perin, M. Berno, T. Erseghe, and M. Rossi, "Towards sustainable edge computing through renewable energy resources and online, distributed and predictive scheduling," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 306–321, 2021.

[31] L. Gu, W. Zhang, Z. Wang, D. Zeng, and H. Jin, "Service management and energy scheduling toward low-carbon edge computing," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 1, pp. 109–119, 2022.

[32] J. Xue and X. Guan, "Collaborative computation offloading and resource allocation based on dynamic pricing in mobile edge computing," *Comput. Commun.*, vol. 198, pp. 52–62, 2023.

[33] B. Shi, Y. Pan, and L. Huang, "Deep reinforcement learning based task offloading and resource allocation strategy across multiple edge servers," *Serv. Orient. Comput. Appl.*, pp. 1–14, 2024.

[34] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, 2022.

[35] M. Li, X. Zhou, T. Qiu, Q. Zhao, and K. Li, "Multi-relay assisted computation offloading for multi-access edge computing systems with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10941–10956, 2021.

[36] J. Zhang, J. Du, Y. Shen, and J. Wang, "Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9303–9317, 2020.

[37] J. Liu, X. Du, J. Cui, M. Pan, and D. Wei, "Task-oriented intelligent networking architecture for the space–air–ground–aqua integrated network," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5345–5358, 2020.

[38] B. Bandyopadhyay, P. Kuila, M. Bey, and M. C. Govil, "Quantum-inspired differential evolution for freshness-aware caching-aided offloading in digital twin-enabled internet of vehicles," *IEEE Trans. Intell. Veh.*, 2024.

[39] M. L. Littman, "Markov games as a framework for multiagent reinforcement learning," in *Proc. Mach. Learn. Proc. 1994*, 1994, pp. 157–163.

[40] M. L. Puterman, "Markov decision processes," *Handb. Oper. Res. Manag. Sci.*, vol. 2, pp. 331–434, 1990.
[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[42] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.
[43] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[44] A. Vaswani, "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[45] X. Han, D. Tian, J. Zhou, X. Duan, Z. Sheng, and V. C. M. Leung, "Privacy-preserving proxy re-encryption with decentralized trust management for MEC-empowered VANETs," *IEEE Trans. Intell. Veh.*, vol. 8, no. 8, pp. 4105–4119, 2023.
[46] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, 2020.
[47] M. Hevesli, A. M. Seid, A. Erbad, and M. Abdallah, "Multi-agent DRL for queue-aware task offloading in hierarchical MEC-enabled air-ground networks," *IEEE Trans. Cogn. Commun. Netw.*, pp. 1–1, 2025.
[48] Y. Chen, Y. Zhang, Y. Wu, L. Qi, X. Chen, and X. Shen, "Joint task scheduling and energy management for heterogeneous mobile edge computing with hybrid energy supply," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8419–8429, 2020.
[49] D. Zhang, Z. Chen, M. K. Awad, N. Zhang, H. Zhou, and X. S. Shen, "Utility-optimal resource management and allocation algorithm for energy harvesting cognitive radio sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3552–3565, Dec. 2016.

QingHua Zhu (M'15–SM'17) received a Ph. D. degree in industrial engineering from the Guangdong University of Technology, Guangzhou, China, in 2013. From 2014 to 2015 and from 2017 to 2018, he was a Visiting Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. He joined Guangdong University of Technology, Guangzhou, China in 2003, and is currently an Associate Professor at the School of Computer Science and Technology. His research interests include intelligent scheduling and optimization, intelligent manufacturing, cloud computing, edge computing, and discrete event systems.

JinYu Liu is currently pursuing the B.Eng. degree in computer science and technology with the School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China. His research interests include reinforcement learning, robot learning, and scheduling optimization.

Yan Hou received the B.S. and M.S. degrees in computer science from Yangtze University, Jingzhou, China, in 1999 and 2002, respectively, and a Ph.D. degree in industrial engineering from the Guangdong University of Technology, Guangzhou, China, in 2016. Since 2002, she has been with the School of Computer Science and Technology at the Guangdong University of Technology where she is currently an Associate Professor. Her current research interests include production scheduling and optimization, information systems, and software engineering

DaWen Lai received the B. S. degree in information and computing science from Wuhan Institute of Technology, Wuhan, China, in 2022. He is currently pursuing his M. Eng. degree in the School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China. His research interests include mobile edge computing, Internet of Vehicle, and deep reinforcement learning.

ZiMing Ou is currently pursuing his B. Eng. degree in computer science and technology from the School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China. His research interest includes privacy protection.