

Article

Master-Refined MAPPO for Long-Term Joint Resource Scheduling in NOMA-MEC Systems

Jianfei Zhang *  and Shangyu Wu

College of Computer and Control Engineering, Qiqihar University, Qiqihar 161003, China;
2024912315@qqhru.edu.cn

* Correspondence: zhangjianfei@qqhru.edu.cn

Abstract

Mobile edge computing (MEC) enables resource-constrained user devices (UDs) to obtain low-latency computing services by offloading computational tasks to the network edge. Non-orthogonal multiple access-enabled mobile edge computing (NOMA-MEC) systems feature asymmetric states across UD, dynamic task arrivals, and competition for wireless and edge computing resources. Under these conditions, offloading decisions affect device energy consumption, task delay, and edge computing resource allocation, making long-term system optimization difficult. This study jointly optimizes task offloading and system resource scheduling to minimize the long-term delay–energy cost. The problem is formulated as a partially observable Markov decision process (POMDP) and addressed using a master-refined multi-agent proximal policy optimization (MR-MAPPO) algorithm. MR-MAPPO combines continuous action relaxation, master action refinement, and a behavior cloning auxiliary term to learn policies in a hybrid discrete–continuous action space. A marginal congestion delay term is also introduced to capture the impact of newly admitted tasks on existing edge workloads. Simulation results show that MR-MAPPO outperforms the considered baselines, while ablation studies verify the effects of its key components. Under the main experimental setting, MR-MAPPO reduces the system cost by 17.9% and 22.9% relative to standard MAPPO and particle swarm optimization (PSO), respectively.

Keywords: mobile edge computing; non-orthogonal multiple access; joint resource scheduling; computation offloading; multi-agent reinforcement learning

1. Introduction

Mobile edge computing (MEC) provides essential support for latency-sensitive and computation-intensive services by deploying computing resources at the network edge [1,2]. With the rapid growth of applications such as the industrial Internet of Things, augmented reality, and collaborative sensing among smart terminals, user devices (UDs) that rely only on local computation can hardly meet the requirements of low latency, low energy consumption, and limited battery capacity at the same time. MEC allows terminals to offload part of their computation tasks to an edge server (ES), thereby alleviating the limitation of local computing capability. However, in dense multi-user access scenarios, both wireless spectrum resources and edge computing resources are scarce. Therefore, coordinated scheduling between communication and computation becomes crucial for improving the performance of MEC systems.

Non-orthogonal multiple access mobile edge computing (NOMA-MEC) provides highly concurrent uplink access for multi-user MEC, but it also makes the coupling between



Received: 23 June 2026

Revised: 16 July 2026

Accepted: 21 July 2026

Published: 22 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

wireless access and edge computing more pronounced [3,4]. Unlike orthogonal access schemes, NOMA allows multiple users to upload tasks over the same time-frequency resource block and relies on successive interference cancellation (SIC) for signal detection. This mechanism improves spectrum efficiency, but the interference structure under concurrent transmission changes dynamically with user access states and power control. Meanwhile, the cache and computing resources of the edge server are limited, and excessive offloading may shift the computation burden from terminals to queuing congestion at the edge [5]. Moreover, UD are naturally asymmetric in task queues, channel gains, battery states, and SIC decoding positions, making a uniform offloading and resource allocation rule difficult to apply to all devices. Therefore, NOMA-MEC systems require coordinated scheduling between wireless access and edge computing according to user-state differences, rather than isolated optimization within a single resource domain.

Dynamic task arrivals and cross-slot state evolution further increase the complexity of MEC scheduling [6,7]. In practical systems, after tasks arrive randomly, they continuously accumulate, are processed, and are transferred across local queues and edge execution queues. The current scheduling action not only affects the immediate service cost, but also changes task backlogs, device energy states, and edge workloads in subsequent slots. Conventional optimization methods usually rely on given static problem instances, complete state information, or deterministic models. Under stochastic channels, dynamic task arrivals, and hybrid discrete–continuous decisions, such methods have difficulty balancing real-time responsiveness and long-term performance.

Deep reinforcement learning (DRL) provides a data-driven approach for cross-slot MEC scheduling. Nevertheless, applying DRL to dynamic NOMA-MEC scheduling remains nontrivial because the decision process involves residual local tasks, existing edge workloads, long-term resource competition, and hybrid discrete–continuous control variables [8–10]. In multi-user scenarios, each UD can only observe partial local information, whereas the system-level performance is jointly determined by coupled offloading requests, ES admission, uplink power control, and computing resource allocation. Therefore, an effective learning-based scheduler must improve policy exploration while preserving physical feasibility under partial observability and coupled action constraints [11–13].

Based on the above analysis, dynamic NOMA-MEC systems require a joint resource scheduling method that can simultaneously handle cross-slot state evolution, communication–computation coupling, and decentralized execution constraints. To this end, this paper proposes a master-refined multi-agent proximal policy optimization (MR-MAPPO) algorithm under the centralized training with decentralized execution (CTDE) framework for multi-user dynamic NOMA-MEC scenarios. The main contributions of this paper are summarized as follows.

- For the cross-slot resource coupling problem in NOMA-MEC systems, a joint scheduling framework for offloading admission and communication–computing resources is established. By characterizing the interactions among task queue evolution, wireless access competition, device energy variation, and edge workload accumulation, a long-term weighted delay–energy cost minimization problem is formulated. Offloading requests, edge admission, computing resource allocation, and uplink transmit power are jointly optimized.
- For concurrent edge execution, a task-level edge workload evolution model is developed, allowing multiple unfinished offloaded tasks from the same UD to coexist at the ES. Furthermore, given the edge active task set, a closed-form solution to the edge central processing unit (CPU) frequency allocation subproblem is derived to characterize computing resource competition among edge tasks.

- To solve the above dynamic joint scheduling problem, an MR-MAPPO algorithm is proposed. Under the CTDE framework, the algorithm combines continuous action relaxation with a master action refinement mechanism. It uses physically feasible refined actions to improve sample quality in the early training stage, and introduces a behavior cloning (BC) auxiliary term to guide the policy toward the refined actions. As training proceeds, the influence of this auxiliary term gradually decreases, allowing policy optimization to return to the proximal policy optimization (PPO) long-term return objective.
- For reward design, an edge marginal congestion delay is introduced to characterize the external impact of newly admitted tasks on existing edge tasks. Simulation results show that MR-MAPPO outperforms the baseline algorithms in system cost, training convergence, and scalability with respect to the number of users. Ablation experiments further indicate that the NOMA access mechanism based on successive interference cancellation (NOMA-SIC), the concurrent edge CPU allocation strategy, and the congestion-aware reward design have a critical impact on system performance.

The remainder of this paper is organized as follows. Section 2 reviews related studies. Section 3 introduces the system model. Section 4 formulates the joint optimization problem. Section 5 presents the MR-MAPPO algorithm. Section 6 provides the performance evaluation. Finally, Section 7 concludes this paper. The key notations used throughout this paper are summarized in Table 1.

Table 1. Key notations.

Notation	Definition
Task Model	
$\mathcal{N}/n/N$	Set/index/number of UDs
$\mathcal{T}/t/\Delta t$	Set/index/duration of decision slots
$\mathcal{J}_{n,t}/L_{n,t}/w_{n,t}$	Head-of-line (HOL) task/input data size/CPU cycles per unit data of the HOL task
$G_{n,t}/g_{n,t}^{\text{rem}}$	Initial/residual computation workload of the HOL task
$u_{n,t}/x_{n,t}$	Offloading request/ES admission decision
$\mathcal{N}_t^{\text{act}}/\mathcal{N}_t^{\text{ES}}$	Active UD set/admitted offloading UD set
Wireless Transmission Model	
$h_{n,t}/p_{n,t}/R_{n,t}$	Channel coefficient/transmit power/uplink rate of UD n
$\mathcal{I}_{n,t}/\text{SINR}_{n,t}$	Residual interfering UD set under SIC/uplink signal-to-interference-plus-noise ratio (SINR)
$T_{n,t}^{\text{tr}}$	Uplink transmission delay
Edge and Local Computing Model	
$f_{n,t}^{\text{loc}}/T_{n,t}^{\text{loc}}$	Local CPU frequency/local execution delay
$\mathcal{Q}_t^-/\mathcal{Q}_t^+$	Existing ES task set before/active edge task set after action execution
$\iota = (n, \tau)/q_{i,t}^-/q_{i,t}^+$	Edge task identifier/residual workload before and after action execution
$F_E/f_{i,t}^-/f_{i,t}^{\text{edge}}$	ES computing capacity/CPU frequency of an existing edge task/CPU frequency used for a newly admitted edge task
$T_{n,t}^{\text{exe}}$	Edge execution delay
$B_t^{\text{ES}}/S^{\text{max}}$	ES cache occupancy/ES cache capacity
Device Energy Consumption and Battery Model	
$E_{n,t}^{\text{tr}}/E_{n,t}^{\text{loc}}/E_{n,t}^{\text{edge}}$	Transmission/local computing/edge computing energy consumption
$b_{n,t}/E^{\text{harv}}/E_{n,t}^{\text{UD}}$	Battery level/harvested energy/UD-side battery energy consumption
System Cost Model	
$\hat{T}_{n,t}/\hat{E}_{n,t}/C_t$	Expected task delay/expected task energy consumption/one-step system cost
$T_{n,t}^{\text{md}}/\mu_{\text{md}}$	Edge marginal congestion delay/weighting coefficient for marginal congestion delay

2. Related Work

Existing MEC studies mainly focus on computation offloading, communication resource allocation, and edge computing resource scheduling. Mach and Becvar [1] systematically review MEC architectures, computation offloading decisions, edge computing resource allocation, and mobility management. Miriyala et al. [14] review deep learning

and deep reinforcement learning methods for computation offloading in edge computing, and compare the applications of DQN, PPO, SAC, and A3C in continuous control, multi-agent coordination, and resource-constrained inference. Lokavarapu et al. [2] analyze computation offloading approaches in multi-access edge computing systems. Zhang et al. [15] study the dynamic offloading problem in MEC with energy-harvesting devices, and formulate the minimization of the weighted sum of energy consumption and execution delay as an online optimization problem with queue stability and battery energy constraints. Verma et al. [6] further review Lyapunov optimization-based task offloading techniques in multi-access edge computing. Dash et al. [16] apply Q-learning to task offloading and resource allocation in dynamic MEC environments to learn long-term reward-driven offloading policies. Aminu et al. [17] use weighted quantum particle swarm optimization for time-energy minimization in MEC, and Wang et al. [7] study sequential computation offloading and migration with multi-task priority.

NOMA-MEC introduces SIC-based non-orthogonal access, creating strong coupling among offloading admission, power control, and edge computing resources. Qiu et al. [3] investigate energy-efficient rate allocation under delay and outage constraints. Shang et al. [8] apply DRL to joint offloading, channel allocation, and transmit power optimization. Wang et al. [9] use DDGP to optimize partial offloading and power allocation under coupled objectives and constraints. Lotfolahi and Ferng [18] employ dual MAPPO to separately handle communication and edge resource allocation in NOMA-aided industrial IoT. Lu et al. [19] develop a decomposition-based multi-agent DRL method for joint resource optimization in STAR-RIS-assisted NOMA-MEC. Wu et al. [20] employ cooperative MATD3 for joint offloading, UAV trajectory, and power allocation in UAV-assisted MEC-NOMA. Zhang et al. [21] develop a shared-network multi-agent DRL method for resource allocation in NOMA-enhanced aerial edge computing. Zhou et al. [22] combine a Multi-Agent Transformer with Lagrangian optimization for offloading and resource allocation in NOMA-based vehicular edge computing. Ahn et al. [4] apply PPO to active RIS-assisted hybrid FDMA-NOMA MEC.

For large-scale and dynamic MEC scenarios, reinforcement learning and multi-agent reinforcement learning are used to reduce the solution complexity of joint scheduling problems. Gao et al. [10] propose the ARMAAC framework, which handles multi-user offloading in large-scale heterogeneous MEC through recurrent actor-critic networks, an attention mechanism, and a centralized critic, while considering dynamic edge resource allocation and task dropping. Liu et al. [11] propose a multi-agent deep Q-network method under the CTDE architecture for sparse code multiple access (SCMA)-MEC networks to jointly optimize SCMA resource allocation and computation offloading. Zhang et al. [23] apply multi-agent deep deterministic policy gradient (MADDPG) to distributed task scheduling in MEC emergency networks. Pan et al. [24] apply MADDPG to energy-optimized computation offloading in MEC-enabled cell-free massive MIMO systems. Wu et al. [12] study multi-agent DRL for computation offloading in cooperative edge networks. Lai et al. [25] use hierarchical multi-agent reinforcement learning for joint computation offloading and resource allocation in LEO satellite networks. Long et al. [26] study reinforcement learning-based fault-tolerant task offloading in MEC.

The above studies provide an important basis for MEC offloading, NOMA access, and multi-agent resource scheduling. Following this line of research, this paper focuses on the joint resource scheduling problem in a single-ES multi-UD dynamic NOMA-MEC scenario, and further characterizes the impact of the time-slot evolution of task queues, edge workloads, and device energy states on scheduling decisions.

3. System Model

As shown in Figure 1, this paper considers an MEC system consisting of one ES and N UD [27]. The UD set is denoted by $\mathcal{N} = \{1, 2, \dots, N\}$. In the physical layer, the MEC system is a continuous-time dynamic system. To enable computationally tractable resource scheduling, system control and scheduling decisions proceed periodically over discrete time slots. Let T denote the total number of slots during system operation. The slot index set is defined as $\mathcal{T} = \{0, 1, \dots, T-1\}$, and the physical duration of each slot is Δt seconds.

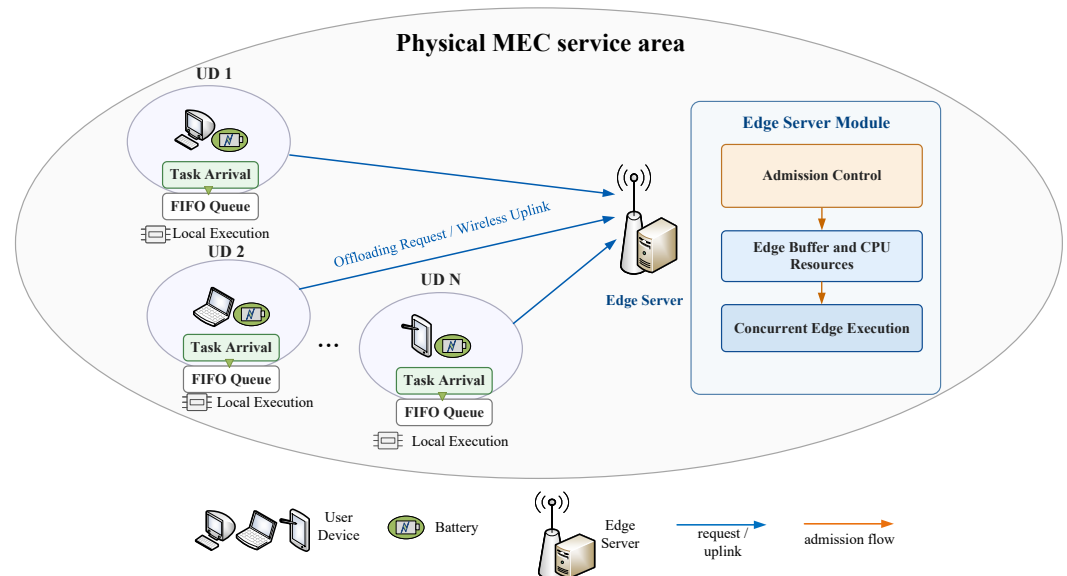


Figure 1. System model of the considered dynamic NOMA-MEC system.

During system operation, each UD dynamically generates computation tasks. In each slot $t \in \mathcal{T}$, UD $n \in \mathcal{N}$ decides whether to submit an offloading request according to its local task attributes and current state. The request variable is denoted by $u_{n,t} \in \{0, 1\}$, where $u_{n,t} = 1$ indicates that UD n requests offloading, and $u_{n,t} = 0$ otherwise. The ES receives the requests and performs admission control. The final execution mode is defined by the admission variable $x_{n,t} \in \{0, 1\}$:

$$x_{n,t} = \begin{cases} 1, & \text{offloaded to the ES,} \\ 0, & \text{executed locally.} \end{cases} \quad (1)$$

Thus, the admission decision must satisfy

$$x_{n,t} \leq u_{n,t}, \quad \forall n \in \mathcal{N}, t \in \mathcal{T}, \quad (2)$$

which means that the ES can admit only tasks that have submitted offloading requests. If UD n has no pending task in slot t , then $u_{n,t} = 0$ is enforced, and the above constraint gives $x_{n,t} = 0$. In this case, $x_{n,t} = 0$ does not indicate local execution, but instead means that no valid offloading admission exists.

3.1. Task Model

During system operation, each UD generates computation tasks according to an independent random process. Newly arrived tasks enter the corresponding local first-in-first-out (FIFO) queue of each UD in order [7,15]. To strictly align the continuous physical process with the discrete control logic, the system takes a state snapshot at the beginning

of each slot $t \in \mathcal{T}$. New tasks arriving during slot $(t, t + 1]$ are observed uniformly in the state update of slot $t + 1$.

In any decision slot $t \in \mathcal{T}$, if the local queue of UD n is nonempty, the current head-of-line (HOL) task is denoted by $\mathcal{J}_{n,t} = (L_{n,t}, w_{n,t})$. Here, $L_{n,t}$ denotes the original input data size of the HOL task when it is generated, measured in bits, and $w_{n,t}$ denotes the number of CPU cycles required to process one unit of data. The initial total computation workload of the HOL task is defined as

$$G_{n,t} = L_{n,t}w_{n,t}. \quad (3)$$

Let $g_{n,t}^{\text{rem}}$ denote the residual computation workload of the HOL task at the beginning of slot t , with $g_{n,t}^{\text{rem}} = G_{n,t}$ for a newly generated task. Before the task is completed and removed from the queue, its static attributes $L_{n,t}$ and $w_{n,t}$ remain strictly unchanged across slots, while its actual execution progress is characterized only by the decreasing update of $g_{n,t}^{\text{rem}}$.

3.2. Wireless Transmission Model

The system synchronously observes the wireless channel state at the beginning of each decision slot $t \in \mathcal{T}$. In NOMA uplink transmission, the ES receiver performs SIC, where users are decoded in descending order of channel gain [3,8,28,29]. Let $\mathcal{N}_t^{\text{ES}} = \{n \in \mathcal{N} \mid x_{n,t} = 1\}$ denote the set of users admitted for offloading by the ES in slot t .

For any admitted user $n \in \mathcal{N}_t^{\text{ES}}$, let $p_{n,t}$ denote its uplink transmit power in slot t , and let $h_{n,t}$ denote its channel coefficient. The wireless channel is modeled by combining distance-dependent path loss and small-scale fading as

$$h_{n,t} = \sqrt{\rho_0 d_n^{-\kappa}} \zeta_{n,t}, \quad (4)$$

where d_n denotes the distance between UD n and the ES, ρ_0 is the reference channel power gain at 1 m, κ is the path-loss exponent, and $\zeta_{n,t}$ denotes the small-scale fading coefficient.

When the ES receiver decodes the signal of user n , the signals of all users with better channel conditions than user n have already been decoded and removed. Thus, user n is interfered with only by users with weaker channels that have not yet been decoded. The set of weak users causing interference to user n is defined as

$$\mathcal{I}_{n,t} = \left\{ j \in \mathcal{N}_t^{\text{ES}} \mid |h_{j,t}|^2 < |h_{n,t}|^2 \right\}. \quad (5)$$

Under the assumption of ideal SIC with no residual interference, the signal-to-interference-plus-noise ratio (SINR) for decoding user n at the ES receiver is

$$\text{SINR}_{n,t} = \frac{p_{n,t}|h_{n,t}|^2}{\sigma^2 + \sum_{j \in \mathcal{I}_{n,t}} p_{j,t}|h_{j,t}|^2}, \quad (6)$$

where σ^2 denotes the additive white Gaussian noise (AWGN) power. Let W denote the wireless channel bandwidth of the system. The uplink transmission rate of user n is given by

$$R_{n,t} = W \log_2(1 + \text{SINR}_{n,t}). \quad (7)$$

Accordingly, the expected uplink transmission delay of the admitted task is defined as

$$T_{n,t}^{\text{tr}} = \frac{L_{n,t}}{R_{n,t}}. \quad (8)$$

The corresponding uplink transmission energy consumption is

$$E_{n,t}^{\text{tr}} = p_{n,t} T_{n,t}^{\text{tr}}. \quad (9)$$

3.3. Edge and Local Computing Model

3.3.1. Local Computing

The active user set with schedulable HOL tasks at the local side at the beginning of slot t is defined as

$$\mathcal{N}_t^{\text{act}} = \{n \in \mathcal{N} \mid g_{n,t}^{\text{rem}} > 0\}. \quad (10)$$

For any $n \in \mathcal{N}_t^{\text{act}}$, when $x_{n,t} = 0$, the task is not admitted for offloading by the ES, and UD n keeps the current HOL task for local execution. Given the allocated local CPU frequency $f_{n,t}^{\text{loc}}$, the expected local delay for completing all residual computation workload $g_{n,t}^{\text{rem}}$ is

$$T_{n,t}^{\text{loc}} = \frac{g_{n,t}^{\text{rem}}}{f_{n,t}^{\text{loc}}}. \quad (11)$$

Due to the finite length of each discrete decision slot Δt , the actual local computation progress of this task within the current slot is

$$\Delta g_{n,t}^{\text{loc}} = \min(g_{n,t}^{\text{rem}}, f_{n,t}^{\text{loc}} \Delta t). \quad (12)$$

If the task remains unfinished after this progress, its residual computation workload is carried over to the next slot as

$$g_{n,t+1}^{\text{rem}} = g_{n,t}^{\text{rem}} - \Delta g_{n,t}^{\text{loc}}. \quad (13)$$

If $\Delta g_{n,t}^{\text{loc}} = g_{n,t}^{\text{rem}}$, the HOL task is completed and removed from the queue, and the next task becomes the new HOL task in a subsequent state snapshot. The actual one-step physical energy consumption within slot t is calculated as

$$E_{n,t}^{\text{loc}} = \kappa_{\text{loc}} \Delta g_{n,t}^{\text{loc}} (f_{n,t}^{\text{loc}})^2. \quad (14)$$

For cost evaluation, the expected total local energy consumption for completing the residual workload at the current local CPU frequency is defined as

$$\hat{E}_{n,t}^{\text{loc}} = \kappa_{\text{loc}} g_{n,t}^{\text{rem}} (f_{n,t}^{\text{loc}})^2, \quad (15)$$

where κ_{loc} is the energy consumption coefficient of the device CPU. The local CPU frequency is strictly constrained by the hardware physical bounds $f_{n,t}^{\text{loc}} \in [f_n^{\text{min}}, f_n^{\text{max}}]$, where f_n^{min} and f_n^{max} denote the minimum and maximum physical execution frequencies supported by the device, respectively.

3.3.2. Edge Concurrent Scheduling

The admission decision of the ES is constrained by the maximum ES cache capacity S^{max} . Let B_t^{ES} denote the cache size occupied by existing ES-side tasks in the snapshot at the beginning of slot t . The newly offloaded tasks in this slot must satisfy the cache constraint based on the initial snapshot:

$$B_t^{\text{ES}} + \sum_{n \in \mathcal{N}_t^{\text{act}}} x_{n,t} L_{n,t} \leq S^{\text{max}}. \quad (16)$$

When $x_{n,t} = 1$ and the above constraint holds, the current HOL task of UD n enters the ES-side cache and computing queue after uplink transmission, and is removed from the local FIFO queue. Since multiple unfinished offloaded tasks from the same UD may

coexist at the ES, this paper uses a tuple $\iota = (n, \tau)$ to identify each ES-side edge task, where n denotes the UD to which the task belongs, and τ denotes the slot in which the task is admitted by the ES.

To characterize edge state evolution under discrete slots, let $\mathcal{Q}_t^-$ denote the set of existing unfinished tasks at the ES at the beginning of slot t . If $\mathcal{Q}_t^- \neq \emptyset$, each existing edge task $\iota \in \mathcal{Q}_t^-$ has a residual computation workload $q_{\iota,t}^-$. Let F_E denote the aggregate computing capacity of the ES. At the task level, the ES computing resource is abstracted as a divisible computing resource pool shared by concurrent edge tasks. The effective computing share of task ι is allocated according to the square-root proportional rule, whose closed-form derivation is provided in Appendix A:

$$f_{\iota,t}^- = F_E \frac{\sqrt{q_{\iota,t}^-}}{\sum_{\ell \in \mathcal{Q}_t^-} \sqrt{q_{\ell,t}^-}}. \quad (17)$$

Within the current slot, the actual computation progress of existing task ι is

$$\Delta q_{\iota,t} = \min(q_{\iota,t}^-, f_{\iota,t}^- \Delta t). \quad (18)$$

After this progress, completed tasks are removed from the ES queue, whereas unfinished tasks remain. Therefore, the edge active task set after action execution is

$$\mathcal{Q}_t^+ = \{\iota \in \mathcal{Q}_t^- \mid q_{\iota,t}^- > \Delta q_{\iota,t}\} \cup \{(n, t) \mid n \in \mathcal{N}_t^{\text{ES}}\}. \quad (19)$$

For any active task ι in $\mathcal{Q}_t^+$, the effective residual computation workload is defined as

$$q_{\iota,t}^+ = \begin{cases} q_{\iota,t}^- - \Delta q_{\iota,t}, & \iota \in \mathcal{Q}_t^-, q_{\iota,t}^- > \Delta q_{\iota,t}, \\ g_{n,t}^{\text{rem}}, & \iota = (n, t), n \in \mathcal{N}_t^{\text{ES}}. \end{cases} \quad (20)$$

For a newly admitted task (n, t) , the edge CPU frequency used for delay and energy evaluation is computed over the updated active task set $\mathcal{Q}_t^+$ as

$$f_{n,t}^{\text{edge}} = F_E \frac{\sqrt{q_{(n,t),t}^+}}{\sum_{\iota \in \mathcal{Q}_t^+} \sqrt{q_{\iota,t}^+}}. \quad (21)$$

Based on the above CPU allocation, the execution delay of the newly admitted task (n, t) at the edge is evaluated as

$$T_{n,t}^{\text{exe}} = \frac{q_{(n,t),t}^+}{f_{n,t}^{\text{edge}}}. \quad (22)$$

Its expected edge computing energy consumption is

$$E_{n,t}^{\text{edge}} = \kappa_{\text{edge}} q_{(n,t),t}^+ (f_{n,t}^{\text{edge}})^2, \quad (23)$$

where κ_{edge} is the CPU energy consumption coefficient at the ES.

3.4. Device Energy Consumption and Battery Model

The system energy consumption evaluation includes three parts: local computation, uplink transmission, and edge execution. However, for the battery state of an individual UD, only the local computation energy consumption and uplink transmission energy

consumption generated by the UD itself reduce its battery level. Therefore, the actual battery energy consumption of UD n in slot t is defined as

$$E_{n,t}^{\text{UD}} = \begin{cases} (1 - x_{n,t})E_{n,t}^{\text{loc}} + x_{n,t}E_{n,t}^{\text{tr}}, & n \in \mathcal{N}_t^{\text{act}}, \\ 0, & n \notin \mathcal{N}_t^{\text{act}}. \end{cases} \quad (24)$$

The device battery state is affected only by its own physical energy consumption and energy harvesting. Therefore, the battery state update of UD n is given by

$$b_{n,t+1} = \min\left(\max\left(b_{n,t} - E_{n,t}^{\text{UD}} + E^{\text{harv}}, 0\right), b^{\text{max}}\right), \quad (25)$$

where E^{harv} denotes the harvested energy in each slot, and b^{max} denotes the battery capacity upper bound [15]. The harvested energy is incorporated into the battery state at the end of the slot and cannot be used to satisfy the physical energy consumption that has already occurred in the current slot. The energy feasibility of the current action is determined by $b_{n,t}$ and $E_{n,t}^{\text{UD}}$.

3.5. System Cost Model

For each active UD in slot t , the expected delay and expected energy consumption of its current HOL task are defined as

$$\hat{T}_{n,t} = (1 - x_{n,t})T_{n,t}^{\text{loc}} + x_{n,t}(T_{n,t}^{\text{tr}} + T_{n,t}^{\text{exe}}), \quad (26)$$

$$\hat{E}_{n,t} = (1 - x_{n,t})\hat{E}_{n,t}^{\text{loc}} + x_{n,t}(E_{n,t}^{\text{tr}} + E_{n,t}^{\text{edge}}). \quad (27)$$

The one-step system cost is defined as the average weighted delay–energy cost over all active UDs:

$$C_t = \begin{cases} \frac{1}{|\mathcal{N}_t^{\text{act}}|} \sum_{n \in \mathcal{N}_t^{\text{act}}} \left(\omega_T \frac{\hat{T}_{n,t}}{T^{\text{ref}}} + \omega_E \frac{\hat{E}_{n,t}}{E^{\text{ref}}} \right), & |\mathcal{N}_t^{\text{act}}| > 0, \\ 0, & |\mathcal{N}_t^{\text{act}}| = 0, \end{cases} \quad (28)$$

where ω_T and ω_E are the delay and energy weights, respectively, satisfying $\omega_T, \omega_E \geq 0$ and $\omega_T + \omega_E = 1$. T^{ref} and E^{ref} are the reference scales for delay and energy, respectively, which map physical quantities with different units to comparable numerical scales for weighted aggregation. For edge tasks admitted in previous slots, their transmission delay, edge execution delay, and edge computing energy consumption have already been included in the system cost at their admission slots. Therefore, these existing tasks are not counted again in subsequent slots; their residual computation workloads only serve as part of the edge workload state for subsequent concurrent CPU allocation.

3.6. Asymmetric System Characteristics

The above system model shows that the considered NOMA-MEC scenario is inherently asymmetric across UDs and time slots. Different UDs may have different task backlogs, channel gains, battery levels, SIC decoding positions, and local computing capabilities, while the ES may maintain uneven residual workloads from previously admitted tasks. As a result, the same offloading, power control, or CPU allocation decision may lead to different delay–energy effects for different UDs. Such asymmetric system states make a uniform scheduling rule inefficient and require an adaptive joint scheduling method that can exploit local state differences while coordinating global resource competition.

4. Problem Formulation

This paper aims to jointly optimize the UD-side offloading request $\mathbf{u}_t = \{u_{n,t}\}_{n \in \mathcal{N}}$, the ES admission decision $\mathbf{x}_t = \{x_{n,t}\}_{n \in \mathcal{N}}$, the local CPU frequency $\mathbf{f}_t = \{f_{n,t}^{\text{loc}}\}_{n \in \mathcal{N}}$, and the uplink transmit power $\mathbf{p}_t = \{p_{n,t}\}_{n \in \mathcal{N}}$, while satisfying the constraints on wireless access, ES cache capacity, device transmit power, local computing frequency, ES computing resources, and battery energy. Let π denote the joint scheduling policy associated with the above control variables.

Based on the discrete-slot decision framework defined above, this paper seeks the optimal policy π that minimizes the expected cumulative system cost over the entire decision horizon $\mathcal{T}$. The joint optimization problem **P** is formulated as

$$\begin{aligned}
 \mathbf{P}: \quad & \min_{\pi} \mathbb{E}_{\pi} \left[\sum_{t \in \mathcal{T}} C_t \right] \\
 & \text{C1: } u_{n,t}, x_{n,t} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}, \\
 & \text{C2: } u_{n,t} = 0, \quad \forall n \notin \mathcal{N}_t^{\text{act}}, \forall t \in \mathcal{T}, \\
 & \text{C3: } x_{n,t} \leq u_{n,t}, \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}, \\
 & \text{C4: } B_t^{\text{ES}} + \sum_{n \in \mathcal{N}_t^{\text{act}}} x_{n,t} L_{n,t} \leq S^{\text{max}}, \quad \forall t \in \mathcal{T}, \\
 & \text{C5: } p_{n,t} \in [p^{\min}, p^{\max}], \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}, \\
 & \text{C6: } f_{n,t}^{\text{loc}} \in [f_n^{\min}, f_n^{\max}], \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}, \\
 & \text{C7: } \sum_{i \in \mathcal{Q}_t^-} f_{i,t}^- \leq F_E, \quad \forall t \in \mathcal{T}, \\
 & \text{C8: } E_{n,t}^{\text{UD}} \leq b_{n,t}, \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}, \\
 & \text{C9: } 0 \leq b_{n,t} \leq b^{\text{max}}, \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}.
 \end{aligned} \tag{29}$$

Here, C1 specifies the binary nature of the offloading request and admission decisions. C2 and C3 describe the sequential admission logic that no task implies no request, and no request implies no admission. C4 limits the ES cache capacity. C5 and C6 constrain the uplink transmit power and local CPU frequency, respectively. C7 represents the ES computing resource capacity constraint for concurrent edge tasks. C8 represents the battery energy feasibility of the current action. C9 gives the boundary of the battery state.

Problem **P** involves both discrete offloading/admission decisions and continuous communication–computing resource control, and is constrained by wireless interference, ES cache capacity, ES computing resources, and device battery energy [30]. Therefore, problem **P** is a mixed-integer nonlinear programming (MINLP) problem. Since the system state evolves continuously with task arrivals, channel variations, battery states, and edge workloads, static optimization methods are difficult to apply directly to this cross-slot stochastic scheduling problem. Therefore, this paper models problem **P** as a partially observable Markov decision process (POMDP) and solves it using a reinforcement learning algorithm under the CTDE framework.

5. Master-Refined MAPPO Algorithm

To solve the above dynamic joint scheduling problem, this paper proposes an MR-MAPPO algorithm under the CTDE framework. This section first formulates the problem as a POMDP, including the state, observation, action, and reward designs. It then presents the master action refinement mechanism and the parameter update process of MR-MAPPO.

5.1. POMDP Formulation

A POMDP is formalized as a tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. In each decision slot $t \in \mathcal{T}$, the environment is in a global state $\mathbf{s}_t \in \mathcal{S}$. Each agent, namely UD n , obtains a local observation vector $\mathbf{o}_{n,t} \in \mathcal{O}$ from the current state and generates an action $\mathbf{a}_{n,t} \in \mathcal{A}$ through its policy network. After receiving the joint actions of all agents, the environment evolves to the next-slot state $\mathbf{s}_{t+1}$ according to the state transition mechanism $\mathcal{P} : \mathcal{S} \times \mathcal{A}^N \rightarrow \mathcal{S}$, and returns a one-step reward $r_{n,t} \in \mathcal{R}$ to each agent. The discount factor $\gamma \in [0, 1)$ balances the importance of future long-term returns against the current immediate reward. The core elements are defined as follows.

5.1.1. Observation and State Spaces

In decision slot t , the local observation vector $\mathbf{o}_{n,t} \in \mathcal{O}$ of UD n consists of the channel condition, device state, task attributes, and edge workload information, and is defined as

$$\mathbf{o}_{n,t} = [h_{n,t}, b_{n,t}, L_{n,t}, w_{n,t}, \mathcal{S}_{n,t}^{\text{rem}}, B_t^{\text{ES}}]. \quad (30)$$

To remove the differences in physical dimensions, all the above state components are strictly normalized before being used as network inputs.

This paper constructs histogram features based on the residual computation workloads of tasks in the edge task set $\mathcal{Q}_t^-$ at the beginning of the slot. Let K_{bin} denote the number of histogram bins. The edge workload feature vector is given by

$$\mathbf{H}_t = [H_t^{(1)}, H_t^{(2)}, \dots, H_t^{(K_{\text{bin}})}]. \quad (31)$$

This feature is used only for centralized value estimation, supplementing the residual workload distribution information that is difficult to capture using only the number of edge tasks and the total workload. During training, the centralized global state $\mathbf{s}_t \in \mathcal{S}$ used by the critic network is the concatenation of all local observations and the edge workload histogram [10]:

$$\mathbf{s}_t = (\mathbf{o}_{1,t}, \dots, \mathbf{o}_{N,t}, \mathbf{H}_t). \quad (32)$$

5.1.2. Action Space

To avoid direct policy-gradient optimization in a hybrid discrete–continuous action space, this paper adopts a continuous action relaxation mechanism [8,9]. The raw actor output of UD n in slot t is defined as the normalized continuous action

$$\mathbf{a}_{n,t}^{\pi} = (\tilde{x}_{n,t}, \tilde{f}_{n,t}, \tilde{p}_{n,t}) \in [-1, 1]^3. \quad (33)$$

Here, $\tilde{x}_{n,t}$, $\tilde{f}_{n,t}$, and $\tilde{p}_{n,t}$ denote the normalized control variables for the offloading tendency, local CPU frequency, and uplink transmit power, respectively, rather than directly executed physical variables. After receiving the actor outputs, the environment decodes these continuous actions into actual scheduling variables. First, the offloading request is jointly determined by the offloading tendency and the local active state:

$$u_{n,t} = \begin{cases} 1, & \tilde{x}_{n,t} \geq 0 \text{ and } n \in \mathcal{N}_t^{\text{act}}, \\ 0, & \text{otherwise.} \end{cases} \quad (34)$$

Then, the local CPU frequency and uplink transmit power are linearly mapped from the normalized control variables to their corresponding physically feasible domains:

$$f_{n,t}^{\text{loc}} = f^{\min} + \frac{\tilde{f}_{n,t} + 1}{2} (f_n^{\max} - f^{\min}), \quad (35)$$

$$p_{n,t} = p^{\min} + \frac{\tilde{p}_{n,t} + 1}{2} (p^{\max} - p^{\min}). \quad (36)$$

After all UDs submit their offloading requests $\mathbf{u}_t = \{u_{n,t}\}_{n \in \mathcal{N}}$, the ES generates the final admission decision $\mathbf{x}_t = \{x_{n,t}\}_{n \in \mathcal{N}}$ according to the rules of concurrent access, ES cache capacity, and transmission feasibility, while ensuring $x_{n,t} \leq u_{n,t}$. Therefore, the actor action space is a continuous relaxed space, whereas the system executes the physical joint action $\mathbf{A}_t^{\text{exe}}$ after action decoding and ES admission correction.

5.1.3. Reward Function Design

The training reward is constructed around the expected service cost in problem **P** [18]. For locally executed tasks, the delay used for training is $T_{n,t}^{\text{loc}}$. For admitted offloaded tasks, an edge marginal congestion delay term is introduced in addition to the transmission delay and edge execution delay:

$$\tilde{T}_{n,t} = (1 - x_{n,t})T_{n,t}^{\text{loc}} + x_{n,t}(T_{n,t}^{\text{tr}} + T_{n,t}^{\text{exe}} + \mu_{\text{md}}T_{n,t}^{\text{md}}), \quad (37)$$

where $T_{n,t}^{\text{md}}$ denotes the marginal congestion delay caused by the newly admitted offloaded task to existing edge tasks, and μ_{md} is the weighting coefficient for the marginal congestion delay term. From the edge CPU allocation rule in (17), for an existing task i that remains unfinished after the computation progress in the current slot, its residual computation workload is $q_{i,t}^- - \Delta q_{i,t}$. When the newly admitted task of UD n is not included in the edge active task set, the estimated completion time of this existing task is

$$T_i^{\text{old}} = \frac{\sqrt{q_{i,t}^- - \Delta q_{i,t}}}{F_E} \sum_{\substack{\ell \in \mathcal{Q}_t^- \\ q_{\ell,t}^- > \Delta q_{\ell,t}}} \sqrt{q_{\ell,t}^- - \Delta q_{\ell,t}}. \quad (38)$$

When the new task of UD n is admitted in slot t , the residual computation workload of the task transmitted from the local side is $g_{n,t}^{\text{rem}}$. After this task joins the edge concurrent task set, the expected completion time of existing task i becomes

$$T_i^{\text{new}} = \frac{\sqrt{q_{i,t}^- - \Delta q_{i,t}}}{F_E} \left(\sum_{\substack{\ell \in \mathcal{Q}_t^- \\ q_{\ell,t}^- > \Delta q_{\ell,t}}} \sqrt{q_{\ell,t}^- - \Delta q_{\ell,t}} + \sqrt{g_{n,t}^{\text{rem}}} \right). \quad (39)$$

By subtracting (38) from (39), the extra completion delay imposed by the newly admitted task of UD n on existing task i is obtained as

$$T_i^{\text{new}} - T_i^{\text{old}} = \frac{\sqrt{q_{i,t}^- - \Delta q_{i,t}} \sqrt{g_{n,t}^{\text{rem}}}}{F_E}. \quad (40)$$

By summing the above extra completion delay over all affected existing edge tasks, the edge marginal congestion delay caused by the newly admitted task of UD n is obtained as

$$T_{n,t}^{\text{md}} = \frac{\sqrt{g_{n,t}^{\text{rem}}}}{F_E} \sum_{\substack{i \in \mathcal{Q}_t^- \\ q_{i,t}^- > \Delta q_{i,t}}} \sqrt{q_{i,t}^- - \Delta q_{i,t}}. \quad (41)$$

This term characterizes the congestion externality imposed by newly admitted tasks on existing edge tasks. Similar reward-shaping designs have been adopted in DRL-based

edge resource management to provide more informative training signals for policy optimization [31]. System performance is evaluated using C_t defined above. For any $n \in \mathcal{N}_t^{\text{act}}$, its training cost $c_{n,t}$ is defined as

$$c_{n,t} = \omega_T \frac{\tilde{T}_{n,t}}{T_{\text{ref}}} + \omega_E \frac{\hat{E}_{n,t}}{E_{\text{ref}}}. \quad (42)$$

The one-step immediate reward of UD n in slot t is defined as

$$r_{n,t} = \begin{cases} -c_{n,t}, & n \in \mathcal{N}_t^{\text{act}}, \\ 0, & n \notin \mathcal{N}_t^{\text{act}}. \end{cases} \quad (43)$$

To promote cooperative scheduling among multiple agents, the system-level global reward $\bar{r}_t$ in slot t is defined as the average of the one-step rewards of active UDs:

$$\bar{r}_t = \begin{cases} \frac{1}{|\mathcal{N}_t^{\text{act}}|} \sum_{n \in \mathcal{N}_t^{\text{act}}} r_{n,t}, & |\mathcal{N}_t^{\text{act}}| > 0, \\ 0, & |\mathcal{N}_t^{\text{act}}| = 0. \end{cases} \quad (44)$$

5.2. Algorithm Design

As shown in Figure 2, the proposed algorithm follows the CTDE framework. In the online interaction stage, UD n generates the raw policy action $\mathbf{a}_{n,t}^\pi$ only from its local observation $\mathbf{o}_{n,t}$ through its actor network. The raw policy actions of all UDs are aggregated on the training side into the joint action $\mathbf{A}_t^\pi$. During centralized training, the critic uses global trajectory information for value estimation, and the advantage function is computed based on generalized advantage estimation (GAE); the actor is updated by combining the PPO objective with an early-stage BC auxiliary term. Unlike standard multi-agent proximal policy optimization (MAPPO), this paper introduces a master action refiner on the centralized training side. According to physical feasibility and one-step cost evaluation, the refiner converts the raw joint action into the actually executed action $\mathbf{A}_t^{\text{exe}}$, after which the MEC physical environment performs state transition, reward evaluation, and trajectory caching. This structure preserves distributed actor inference during execution, while using the global state and physical model during training to assist sample generation and parameter updates. The master action refinement mechanism and the parameter update process are presented below.

5.2.1. Master Action Refinement Module

The master action refinement module is used to alleviate inefficient exploration caused by random policies in the early training stage. Specifically, this paper uses heuristic rules to guide and correct the joint actions output by the actors during training.

In each decision slot t , the master first aggregates the raw network actions of all UDs, denoted by

$$\mathbf{A}_t^\pi = \{\mathbf{a}_{1,t}^\pi, \dots, \mathbf{a}_{N,t}^\pi\}. \quad (45)$$

Then, based on the current system state and physical feasibility rules, the master constructs a candidate action set:

$$\mathcal{D}_t^{\text{cand}} = \{\mathbf{A}_t^\pi, \mathbf{A}_t^{\text{loc}}, \mathbf{A}_t^{\text{edge}}, \mathbf{A}_t^{\text{cost}}\}, \quad (46)$$

where $\mathbf{A}_t^{\text{loc}}$ denotes the conservative local execution action, $\mathbf{A}_t^{\text{edge}}$ denotes the edge of-flooding action, and $\mathbf{A}_t^{\text{cost}}$ denotes a low-cost heuristic action constructed from the current cost estimate.

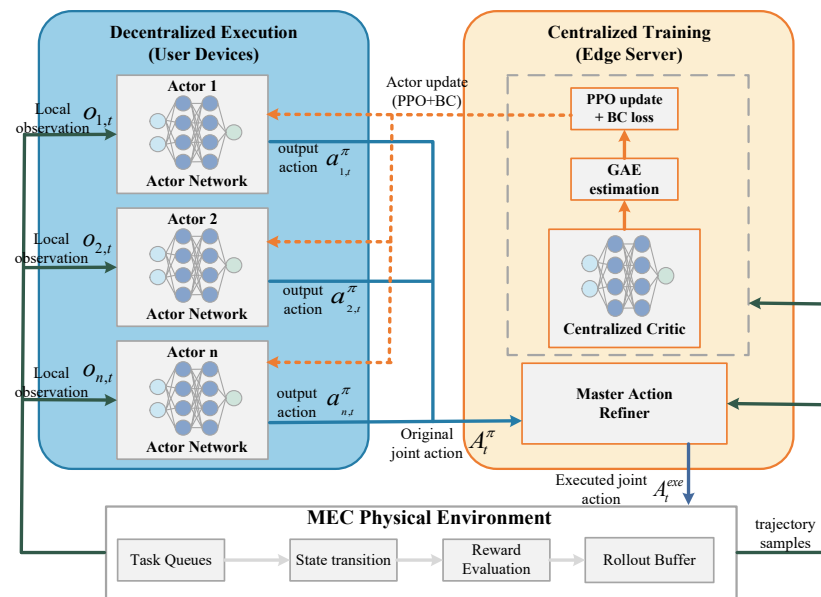


Figure 2. Framework of the proposed MR-MAPPO algorithm under the CTDE paradigm.

The master refiner uses a one-step physical model to quickly evaluate any candidate action $\mathbf{A} \in \mathcal{D}_t^{\text{cand}}$. Let $\hat{r}_t(\mathbf{s}_t, \mathbf{A})$ denote the estimated global reward corresponding to candidate action $\mathbf{A}$ under state $\mathbf{s}_t$. By traversing this finite candidate set, the master directly selects the joint action with the maximum expected return as the refinement reference:

$$\mathbf{A}_t^* = \arg \max_{\mathbf{A} \in \mathcal{D}_t^{\text{cand}}} \hat{r}_t(\mathbf{s}_t, \mathbf{A}). \quad (47)$$

To use master-refined actions to stabilize the hybrid action space in the early training stage, while ensuring that policy updates in the later stage are dominated by the PPO long-term return objective, this paper introduces a decaying training-side guidance probability that decreases with the training episode m :

$$\beta_m = \beta_0 \cdot \max\left(1 - \frac{m}{M_{\text{BC}}}, 0\right), \quad (48)$$

where $\beta_0 \in (0, 1]$ is the initial guidance probability, and M_{BC} is the guidance decay horizon. Let the training-side gating variable be $z_{m,t} \sim \text{Bernoulli}(\beta_m)$. The actually executed joint action in the current slot is defined as

$$\mathbf{A}_t^{\text{exe}} = \begin{cases} \mathbf{A}_t^*, & z_{m,t} = 1, \\ \mathbf{A}_t^\pi, & z_{m,t} = 0. \end{cases} \quad (49)$$

Thus, master-refined actions are used more frequently in the early training stage to provide training guidance and BC targets. As β_m decays, the probability of executing the master-refined action decreases, and actor-generated actions gradually dominate the interaction process, while the PPO objective remains active throughout training.

The transition tuple stored after one-step interaction is given by

$$(\mathbf{s}_t, \{\mathbf{o}_{n,t}\}_{n \in \mathcal{N}}, \mathbf{A}_t^\pi, \mathbf{A}_t^*, \mathbf{A}_t^{\text{exe}}, z_{m,t}, \bar{r}_t, \mathbf{s}_{t+1}). \quad (50)$$

This tuple is used for subsequent advantage estimation and policy updates. In particular, $\mathbf{A}_t^{\text{exe}}$ denotes the action applied to the environment to obtain $\bar{r}_t$ and $\mathbf{s}_{t+1}$, while $\mathbf{A}_t^\pi$ and $\mathbf{A}_t^*$ are retained to identify the raw actor output and the BC target.

5.2.2. Advantage Estimation and PPO Update

After data sampling over one rollout period, the system performs centralized parameter updates based on the sampled trajectories. Let θ denote the current actor parameters, and let θ_{old} denote the old policy parameters used during sampling. Let $\mathbf{a}_{n,t}^{\text{exe}}$ denote the component of $\mathbf{A}_t^{\text{exe}}$ corresponding to UD n . For this actually executed action, the PPO probability ratio is defined as

$$\rho_{n,t}(\theta) = \frac{\pi_{\theta}(\mathbf{a}_{n,t}^{\text{exe}} | \mathbf{o}_{n,t})}{\pi_{\theta_{\text{old}}}(\mathbf{a}_{n,t}^{\text{exe}} | \mathbf{o}_{n,t})}. \quad (51)$$

The actor objective consists of the PPO clipped surrogate term [32], policy entropy regularization, and an early-stage BC auxiliary term. The single-sample PPO clipped objective is defined as

$$\mathcal{J}_{n,t}^{\text{PPO}}(\theta) = \min(\rho_{n,t}(\theta)\hat{\mathcal{A}}_t, \text{clip}(\rho_{n,t}(\theta), 1 - \epsilon, 1 + \epsilon)\hat{\mathcal{A}}_t). \quad (52)$$

Combined with the gating variable $z_{m,t}$, the actor objective is written as

$$\begin{aligned} \mathcal{L}^{\text{actor}}(\theta) = \mathbb{E}_{t,n} \Big[& \mathcal{J}_{n,t}^{\text{PPO}}(\theta) - z_{m,t}\eta_{\text{BC}} \|\boldsymbol{\mu}_{\theta}(\mathbf{o}_{n,t}) - \mathbf{a}_{n,t}^{\star}\|_2^2 \\ & + \eta_{\text{ent}} \mathcal{H}(\pi_{\theta}(\cdot | \mathbf{o}_{n,t})) \Big]. \end{aligned} \quad (53)$$

Here, $\text{clip}(\cdot)$ is the clipping function, and ϵ is the PPO clipping coefficient. $\mathcal{H}(\cdot)$ denotes the policy entropy, and η_{ent} is the entropy coefficient used to encourage agent exploration. $\boldsymbol{\mu}_{\theta}(\mathbf{o}_{n,t})$ denotes the mean of the current actor policy distribution, $\mathbf{a}_{n,t}^{\star}$ denotes the component of $\mathbf{A}_t^{\star}$ corresponding to UD n , η_{BC} is the BC weight, and $\|\cdot\|_2^2$ denotes the squared ℓ_2 norm, i.e., the squared Euclidean distance. The BC term encourages the actor output to approach the refined action when $z_{m,t} = 1$. As β_m decreases, this guidance gradually weakens and the policy update becomes dominated by the PPO long-term return objective.

In the above objective function, $\hat{\mathcal{A}}_t$ is the advantage function, which measures whether an action is better or worse than the average level. This paper adopts GAE to compute this term [33]:

$$\hat{\mathcal{A}}_t = \sum_{l=0}^{K-t-1} (\gamma \lambda_{\text{GAE}})^l \delta_{t+l}, \quad (54)$$

where l is the temporal offset index in the GAE summation, and $\lambda_{\text{GAE}} \in [0, 1]$ is the GAE smoothing factor. In long-horizon delay and energy optimization, a larger λ_{GAE} reduces bias but increases variance, whereas a smaller λ_{GAE} has the opposite effect. δ_t is the temporal-difference (TD) residual, which measures the accuracy of the critic value prediction for the current state. To keep the target estimate fixed during the current update, the TD residual is computed using the pre-update critic parameters:

$$\delta_t = \bar{r}_t + \gamma V_{\phi_{\text{old}}}(\mathbf{s}_{t+1}) - V_{\phi_{\text{old}}}(\mathbf{s}_t), \quad (55)$$

where $V_{\phi_{\text{old}}}(\cdot)$ denotes the centralized critic value function before the update. Based on the above advantage estimate, the regression target for critic learning is defined as

$$\hat{G}_t = \hat{\mathcal{A}}_t + V_{\phi_{\text{old}}}(\mathbf{s}_t). \quad (56)$$

The critic loss is defined as the mean squared error (MSE) between the current value prediction and the fixed target return:

$$\mathcal{L}^{\text{critic}}(\phi) = \mathbb{E}_t \left[(V_\phi(\mathbf{s}_t) - \hat{G}_t)^2 \right], \quad (57)$$

where $\mathbb{E}_t$ denotes the empirical average over all time steps in the rollout trajectory.

Finally, the system updates the actor parameters by gradient ascent and the critic parameters by gradient descent:

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \mathcal{L}^{\text{actor}}(\theta), \quad (58)$$

$$\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \mathcal{L}^{\text{critic}}(\phi), \quad (59)$$

where α_θ and α_ϕ are the learning rates.

Algorithm 1 summarizes the master action refinement operation in one decision slot.

Algorithm 1: Master action refinement in one decision slot.

Input: Current state $\mathbf{s}_t$, raw joint actor action $\mathbf{A}_t^\pi$.

Output: Refined joint action $\mathbf{A}_t^*$.

- 1: Set $\mathbf{A}_t^{\text{loc}}$ as a conservative fallback candidate by assigning active UD to local execution with feasible local CPU controls.
 - 2: Set $\mathbf{A}_t^{\text{edge}}$ as an edge offloading candidate by giving offloading priority to active UD with favorable channel and access conditions.
 - 3: Set $\mathbf{A}_t^{\text{cost}}$ as a cost-aware candidate by ranking active UD with the one-step cost model and prioritizing UD with larger estimated cost reduction.
 - 4: Construct $\mathcal{D}_t^{\text{cand}} \leftarrow \{\mathbf{A}_t^\pi, \mathbf{A}_t^{\text{loc}}, \mathbf{A}_t^{\text{edge}}, \mathbf{A}_t^{\text{cost}}\}$.
 - 5: **for each** $\mathbf{A} \in \mathcal{D}_t^{\text{cand}}$ **do**
 - 6: Decode $\mathbf{A}$ into $\mathbf{u}_t$, $\mathbf{f}_t^{\text{loc}}$, and $\mathbf{p}_t$ according to (34)–(36).
 - 7: **for each** UD n with $u_{n,t} = 1$ **do**
 - 8: **if** admitting UD n keeps C2–C4 in problem $\mathbf{P}$ satisfied and the decoded candidate satisfies C5–C8 **then**
 - 9: Set $x_{n,t} = 1$.
 - 10: **else**
 - 11: Set $x_{n,t} = 0$ and use local fallback.
 - 12: **end if**
 - 13: **end for**
 - 14: Estimate $\hat{r}_t(\mathbf{s}_t, \mathbf{A})$ using the one-step reward model in (42)–(44).
 - 15: **end for**
 - 16: Return $\mathbf{A}_t^* = \arg \max_{\mathbf{A} \in \mathcal{D}_t^{\text{cand}}} \hat{r}_t(\mathbf{s}_t, \mathbf{A})$.
-

Algorithm 2 summarizes the training procedure of the proposed MR-MAPPO joint resource scheduling algorithm. Under the CTDE framework, the actor of each UD generates a raw action from its local observation, while the training side uses the master action refinement mechanism to determine the actually executed action and stores the resulting interaction samples for PPO updates and BC auxiliary learning. After training, only the distributed actor networks are retained for online execution, enabling each UD to make low-latency decisions based on local observations.

Algorithm 2: Training procedure of MR-MAPPO.**Input:** $M, K, \gamma, \lambda_{\text{GAE}}, \epsilon, \eta_{\text{ent}}, \alpha_{\theta}, \alpha_{\phi}, \beta_0, M_{\text{BC}}, \eta_{\text{BC}}$.**Output:** Distributed actor policy π_{θ} .

```

1: Initialize actor parameters  $\theta$ , critic parameters  $\phi$ , and set  $\theta_{\text{old}} \leftarrow \theta, \phi_{\text{old}} \leftarrow \phi$ .
2: for training episode  $m = 1$  to  $M$  do
3:   Reset the system environment, task queues, edge workload state, and battery states.
4:   Compute  $\beta_m$  according to (48).
5:   for decision slot  $t = 0$  to  $K - 1$  do
6:     Each UD  $n$  observes  $\mathbf{o}_{n,t}$  and samples  $\mathbf{a}_{n,t}^{\pi} \sim \pi_{\theta_{\text{old}}}(\cdot \mid \mathbf{o}_{n,t})$ .
7:     Aggregate all raw actions into  $\mathbf{A}_t^{\pi}$ .
8:     Run Algorithm 1 with  $(\mathbf{s}_t, \mathbf{A}_t^{\pi})$  to obtain  $\mathbf{A}_t^*$ .
9:     Sample  $z_{m,t} \sim \text{Bernoulli}(\beta_m)$ , and determine  $\mathbf{A}_t^{\text{exe}}$  by (49).
10:    Execute  $\mathbf{A}_t^{\text{exe}}$ , observe  $\bar{r}_t$  and  $\mathbf{s}_{t+1}$ .
11:    Store the transition tuple in (50) into the trajectory buffer.
12:  end for
13:  Compute  $\delta_t, \hat{\mathcal{A}}_t$ , and  $\hat{G}_t$  for all sampled slots using (55), (54), and (56).
14:  Update  $\theta$  and  $\phi$  according to (58) and (59), respectively.
15:  Synchronize  $\theta_{\text{old}} \leftarrow \theta, \phi_{\text{old}} \leftarrow \phi$ .
16: end for
17: return  $\pi_{\theta}$ .

```

As shown in Algorithm 2, each training episode contains K interaction slots. In each slot, distributed action generation by all UD's has complexity $O(N)$. The master refiner in Algorithm 1 traverses the candidate action set $\mathcal{D}_t^{\text{cand}}$ and performs one physical-model evaluation for each candidate joint action, whose complexity can be approximated as $O(N^2)$. Therefore, the action refinement complexity in a single slot is $O(|\mathcal{D}_t^{\text{cand}}|N^2)$. Since the candidate action set has a fixed and small size, $|\mathcal{D}_t^{\text{cand}}|$ can be treated as a constant, and the rollout complexity of each episode is approximately $O(KN^2)$. Considering M training episodes, the overall rollout complexity of MR-MAPPO can be approximated as $O(MKN^2)$. This complexity grows polynomially with the number of users, avoiding the exponential combinatorial search cost caused by directly solving the original MINLP problem.

6. Simulation Results and Discussion

This section presents the simulation settings, baseline algorithms, and performance analysis to verify the long-term scheduling performance of the proposed MR-MAPPO algorithm and the effectiveness of its key components in a dynamic NOMA-MEC environment.

6.1. Simulation Settings

The simulations were conducted on a workstation equipped with an Intel Core i9-12900H processor (Intel Corporation, Santa Clara, CA, USA), 32 GB of memory, and an NVIDIA GeForce RTX 3070 Ti graphics processing unit (GPU; NVIDIA Corporation, Santa Clara, CA, USA). The software environment is based on Python 3.9. PyTorch 2.3.1 was used to build the neural networks, and OpenAI Gym 0.26.1 was used as the reinforcement learning interaction interface.

This paper considers a single-cell NOMA-MEC scenario consisting of $N = 30$ UD's and one ES. The ES is located at the center of a $50 \text{ m} \times 50 \text{ m}$ coverage area, and the UD's are independently and uniformly distributed over this area at the beginning of each training or

evaluation episode and remain stationary within that episode. The channel power gain at 1 m is set to $\rho_0 = -30$ dB, and the path-loss exponent is set to $\kappa = 4$. The small-scale fading is modeled as Rayleigh fading, where $\xi_{n,t} \sim \mathcal{CN}(0, 1)$ is independently generated for each UD at the beginning of each decision slot and remains constant within that slot. The system bandwidth is $W = 30$ MHz, the noise power is $\sigma^2 = -100$ dBm, the decision slot length is $\Delta t = 1$ s, and the uplink transmit power ranges from 14 to 24 dBm.

Each UD generates a new task in each slot with probability $\lambda_n = 0.3$. The task data size follows a truncated exponential distribution, and the computation intensity is uniformly sampled from $[0.5, 2.0]$ Gcycles/MB. The maximum local CPU frequency of each UD follows $\mathcal{U}[0.1, 1.0]$ GHz, and the battery capacity is $b^{\max} = 100$ J. The total computing capacity of the ES is $F_E = 15$ GHz, and the ES cache capacity is $S^{\max} = 100$ MB.

For the proposed MR-MAPPO algorithm, both the actor and critic networks adopt a two-layer multilayer perceptron structure, and each layer contains 128 hidden units. The training process contains 3000 episodes, and each episode consists of 20 decision slots. The Adam optimizer is used, with learning rates of 3×10^{-4} and 10^{-3} for the actor and critic, respectively. For PPO implementation, N_{ppo} denotes the number of PPO optimization epochs for each trajectory buffer, and N_{mb} denotes the number of mini-batches used in each PPO update. The detailed simulation parameters are summarized in Table 2.

Table 2. Simulation parameters.

Parameter	Value	Parameter	Value
Communication			
N	30	W	30 MHz
Coverage area	50 m $\times$ 50 m	κ	4
$\xi_{n,t}$	$\mathcal{CN}(0, 1)$	σ^2	−100 dBm
Δt	1 s	$p_{n,t}$	[14, 24] dBm
Task and computation			
λ_n	0.3	$L_{n,t}$	mean 1.5 MB, range [0.5, 5.0] MB
$w_{n,t}$	$\mathcal{U}[0.5, 2.0]$ Gcycles/MB	$f_n^{\max}$	$\mathcal{U}[0.1, 1.0]$ GHz
F_E	15 GHz	$S^{\max}$	100 MB
Energy and cost			
$b^{\max}$	100 J	$b_{n,0}$	$\mathcal{U}[30, 100]$ J
E^{harv}	0.5 J/slot	κ_{loc}	5×10^{-27}
κ_{edge}	10^{-28}	ω_T	0.5
ω_E	0.5	μ_{md}	1.5
T^{ref}	5.0 s	E^{ref}	2.5 J
Learning			
d_{hid}	128	L_{hid}	2
$\log \sigma_0$	−1.4	α_θ	3×10^{-4}
α_ϕ	10^{-3}	γ	0.99
λ_{GAE}	0.95	ϵ	0.2
η_{ent}	0.001	η_{BC}	1.0
β_0	1.0	M_{BC}	600 episodes
N_{ppo}	5	N_{mb}	4

6.2. Baseline Algorithms

To verify the effectiveness of the proposed MR-MAPPO algorithm and evaluate the contributions of key designs such as master action refinement, centralized value estimation, and long-term return learning, this paper compares it with the following seven baseline algorithms.

- MAPPO [34]: This baseline adopts a multi-agent proximal policy optimization framework. Under the CTDE structure, it uses a centralized critic for value estimation, while the distributed actor of each UD generates actions based on local observations. This baseline does not introduce the master action refiner or the BC auxiliary term, and is

used to evaluate the impact of the master action refinement mechanism on policy search efficiency and system cost.

- MATD3 [19,20]: This baseline adopts a multi-agent twin delayed deep deterministic policy gradient (MATD3) framework, which uses twin critics, target policy smoothing, and delayed policy updates to mitigate value overestimation in deterministic policy gradients. It is used to compare the stability difference between stochastic policy optimization and improved deterministic policy optimization in the hybrid discrete–continuous scheduling problem.
- MADDPG [23,24]: This baseline adopts a multi-agent deep deterministic policy gradient framework. Under the CTDE structure, it uses a centralized critic to evaluate the global state and joint action, while the actor of each UD generates continuous actions based on local observations. This baseline is used to evaluate the scheduling performance of classical deterministic actor–critic methods under the coupled conditions of NOMA interference and shared edge resources.
- Greedy-Cost: This is a one-step physical-cost greedy policy. It estimates the immediate costs of local execution and offloading execution based on the current task, channel, cache, and edge workload states, and preferentially admits offloaded tasks with lower estimated costs under the cache constraint. This baseline is used to measure the benefit of long-term reinforcement learning over myopic scheduling.
- Particle Swarm Optimization (PSO) [35,36]: This population-based iterative optimizer searches in each decision slot for a joint action comprising task offloading, local CPU frequency, and uplink transmit power controls, and evaluates each candidate according to the current-slot system cost. The swarm contains 20 particles and performs 30 iterations per decision slot. The inertia weight decreases linearly from 0.9 to 0.4, while the cognitive and social acceleration coefficients are both set to 1.5. This baseline is used to compare traditional model-driven iterative optimization with the proposed long-term policy-learning method under the same physical constraints.
- Edge-All: This is an aggressive offloading policy. All UDs with pending tasks submit offloading requests and use the maximum uplink transmit power. The final admission decisions are still made by the ES according to ES cache capacity and physical feasibility constraints. This baseline is used to characterize the performance degradation of indiscriminate offloading under NOMA interference and edge congestion.
- Random: This is a stateless random policy. It independently and uniformly samples the offloading tendency, local CPU control variable, and transmit power control variable within the legal action ranges, without using task, channel, or cache state information. This baseline provides a reference for stateless decision making.

6.3. Simulation Results

This subsection evaluates the performance of MR-MAPPO in terms of training convergence, system cost, delay–energy decomposition, scalability with the number of users, sensitivity to edge computing capacity and master-refiner configurations, and ablation of key components. To reduce the influence of randomness caused by network initialization, task arrivals, and channel sampling, MR-MAPPO was independently trained and evaluated under random seeds 37, 42, and 100. The main learning-based comparisons were averaged over three independent random seeds. For the bar-chart results, bar heights denote mean values over repeated evaluation episodes, and error bars denote one standard deviation across those episodes.

Figure 3 shows the convergence curves of the average episode reward for different deep reinforcement learning algorithms during training. MR-MAPPO enters a stable plateau after about 1000 episodes and maintains a reward level of approximately -7.2 in

later training. In contrast, MAPPO converges more slowly, and its average reward after 3000 episodes remains around -9.6 . This indicates that stochastic policy exploration without master action refinement has difficulty quickly locating high-quality action regions in a hybrid action space jointly constrained by ES cache capacity, transmit power, battery energy, and edge concurrent scheduling. The performance gain of MR-MAPPO mainly comes from the model-assisted screening of candidate joint actions by the master refinement mechanism. During training, this mechanism reduces the interference of low-quality offloading decisions and invalid resource configurations in policy updates, thereby improving early-stage sample efficiency and final policy quality. MATD3 and MADDPG also improve during training, but their final rewards stabilize around -11.7 and -11.9 , respectively, which are clearly lower than that of MR-MAPPO. This result shows that, in scenarios with coupled discrete offloading requests, continuous power control, and NOMA interference, deterministic policy-gradient methods are more affected by action-space nonconvexity and multi-agent nonstationarity. Overall, MR-MAPPO shows better training behavior in terms of convergence speed, final reward, and cross-seed stability.

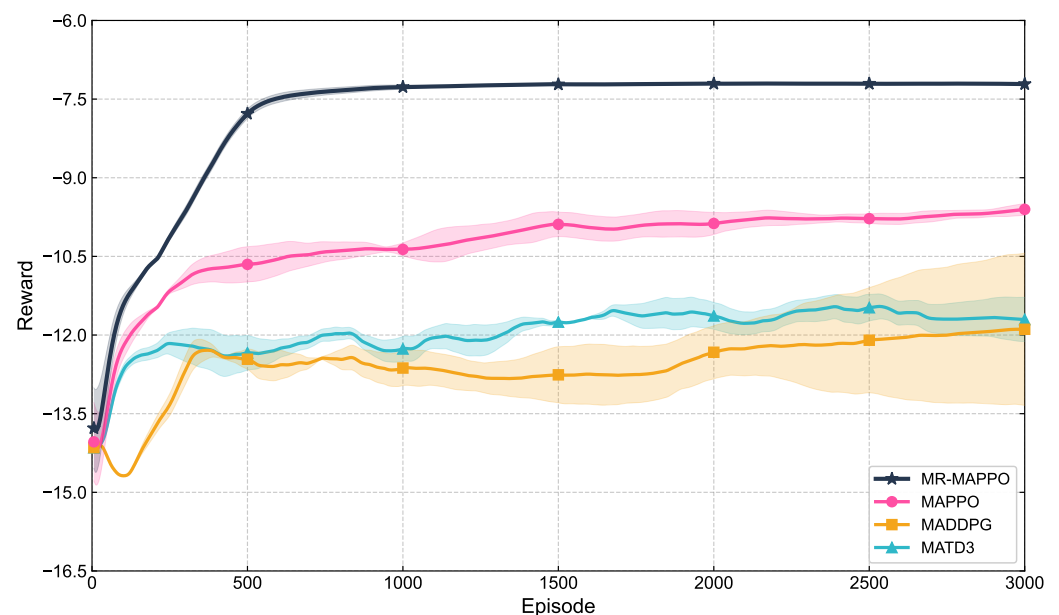


Figure 3. Training convergence comparison. Solid curves denote smoothed mean rewards, and shaded bands show one standard deviation around the mean across three random seeds.

Figure 4 presents the system cost of different algorithms. MR-MAPPO achieves the lowest average system cost over active devices among all compared methods. Its system cost is 7.79, which is 22.9% lower than that of PSO, and it clearly outperforms MAPPO without the master refinement mechanism. Although PSO improves upon Greedy-Cost through population-based iterative search in each decision slot, both methods evaluate actions mainly according to current-slot information and do not learn the long-term consequences of queue evolution and residual edge workloads. The comparison with PSO therefore indicates that the advantage of MR-MAPPO does not arise merely from a larger one-step action-search budget, but from learning scheduling decisions that account for long-term system evolution. Edge-All pushes the workload aggressively to the ES, making edge computing delay the main bottleneck. Random lacks state awareness and thus shows unstable behavior in both delay and energy consumption.

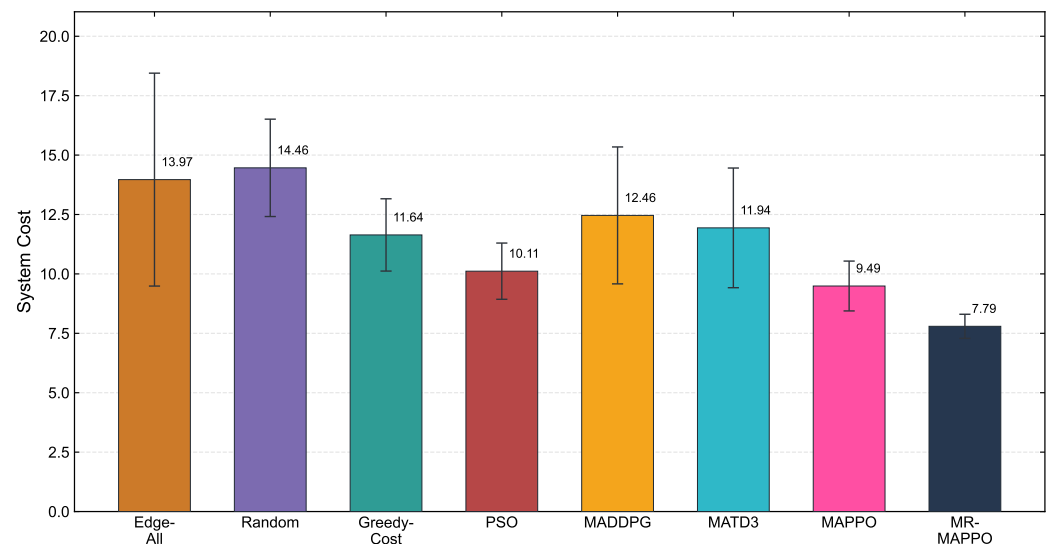


Figure 4. System cost comparison.

The delay and energy decomposition in Figure 5 shows that the advantage of MR-MAPPO mainly lies in service-delay control and system-cost balance, rather than simply pursuing the lowest energy consumption. PSO achieves an average energy consumption of 57.12 J, lower than the 76.59 J obtained by MR-MAPPO, but its average service delay increases from 2.34 s to 2.60 s and its overall system cost remains higher. MATD3 exhibits a similar tendency toward lower energy consumption at the expense of increased delay and system cost. These results indicate that reducing energy consumption alone does not ensure a better solution under the delay–energy weighted objective. By coordinating offloading admission, transmit power, and local CPU frequency, MR-MAPPO reduces the imbalance between edge congestion and local execution burden, thereby achieving better performance under the delay–energy weighted objective defined in this paper. The delay decomposition in Figure 5 is obtained by weighting the average transmission delay, edge computing delay, and local computing delay according to the admission ratio.

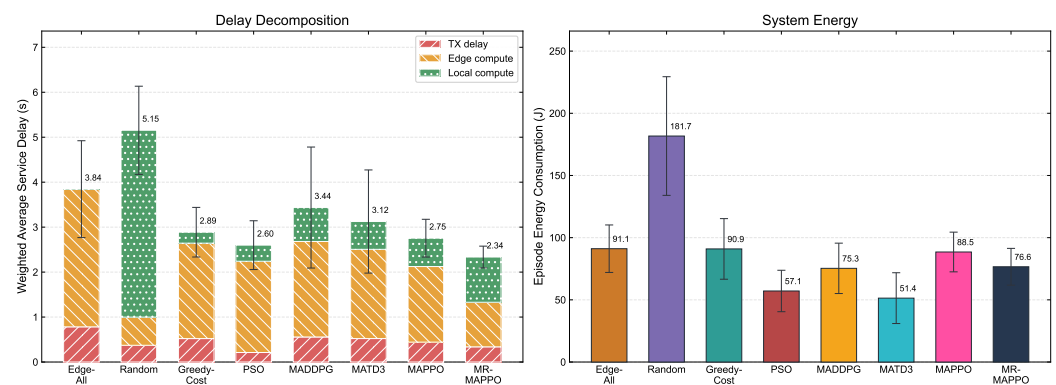


Figure 5. Delay and energy decomposition.

Figure 6 shows the system cost of different algorithms as the number of UDs N changes. As N increases from 20 to 60, the costs of all algorithms increase because the task arrival density, NOMA interference intensity, and ES-side concurrent computing load all increase. MR-MAPPO consistently maintains the lowest system cost, reducing it by approximately 13.1%–27.3% relative to PSO across the evaluated network sizes. MAPPO still clearly outperforms deterministic policy-gradient methods in large-scale scenarios, but its cost increases faster than that of MR-MAPPO. This suggests that relying only on stochastic

policy exploration and centralized value estimation is still insufficient for handling a high-dimensional coupled action space. Edge-All and Random degrade most significantly as the number of users increases, showing that indiscriminate offloading and stateless random decisions cannot adapt to communication interference and edge congestion caused by multi-user access. Greedy-Cost is competitive at a medium scale, but because it selects actions only according to the one-step cost of the current slot, its scalability remains weaker than that of MR-MAPPO with long-term return learning.

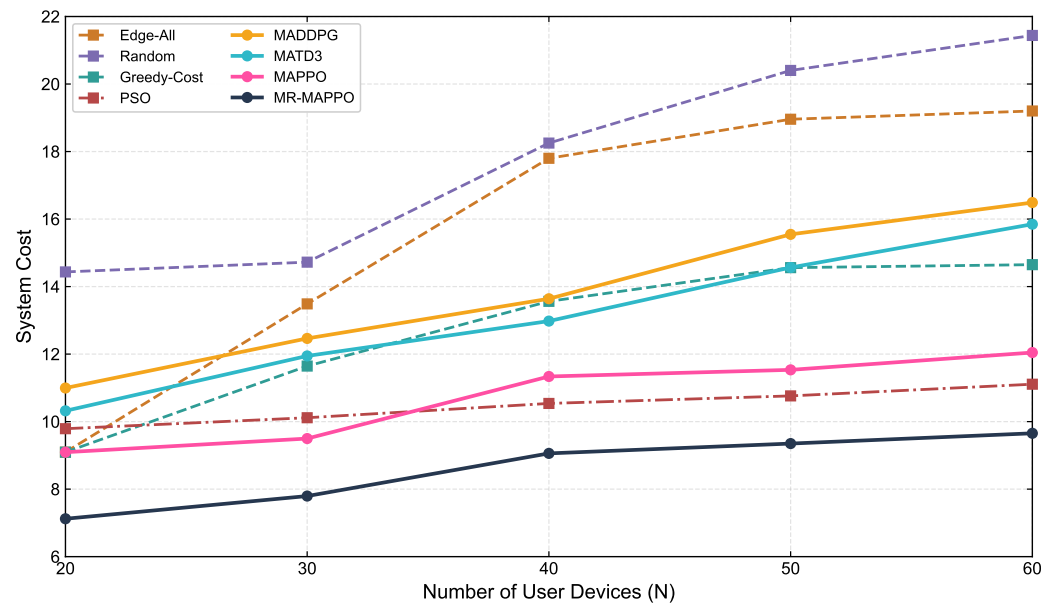


Figure 6. System cost versus number of user devices.

Figure 7 shows the system cost as the edge computing capacity F_E changes. As F_E increases from 5 GHz to 20 GHz, the costs of all algorithms generally decrease, indicating that stronger edge computing capacity can alleviate the ES-side concurrent computing bottleneck. However, when F_E continues to increase, the marginal gains of most methods gradually decrease. This shows that system performance is no longer determined only by edge computing capacity, but is also affected by offloading admission, NOMA transmission interference, and terminal energy constraints. PSO also benefits from the increased ES computing capacity, but its system cost remains higher than that of MR-MAPPO under all evaluated F_E settings. In contrast, Edge-All, Random, MATD3, and MADDPG have higher costs at $F_E = 5$ GHz, reflecting that indiscriminate offloading, stateless exploration, or deterministic policies are more likely to produce inefficient admission and queue backlogs under resource bottlenecks. Greedy-Cost improves significantly as F_E increases, but its later cost reduction slows down, suggesting that one-step cost greediness can exploit the gain from edge computing capacity but still lacks global scheduling ability for long-term queue states and multi-user coupling effects.

Table 3 reports the runtime and memory overhead of MR-MAPPO under different numbers of UD. The reported training time covers 1000 environment interaction steps and the corresponding policy updates. The actor-only decision time denotes the deployment-time inference overhead, where the trained actor maps the current UD observations to scheduling actions without using the centralized critic or the master action refiner. As N increases from 20 to 60, the training time increases due to the larger joint state-action space and the candidate evaluation in the training-side refiner. In contrast, the actor-only decision time remains at the millisecond level, and the peak memory usage changes only slightly.

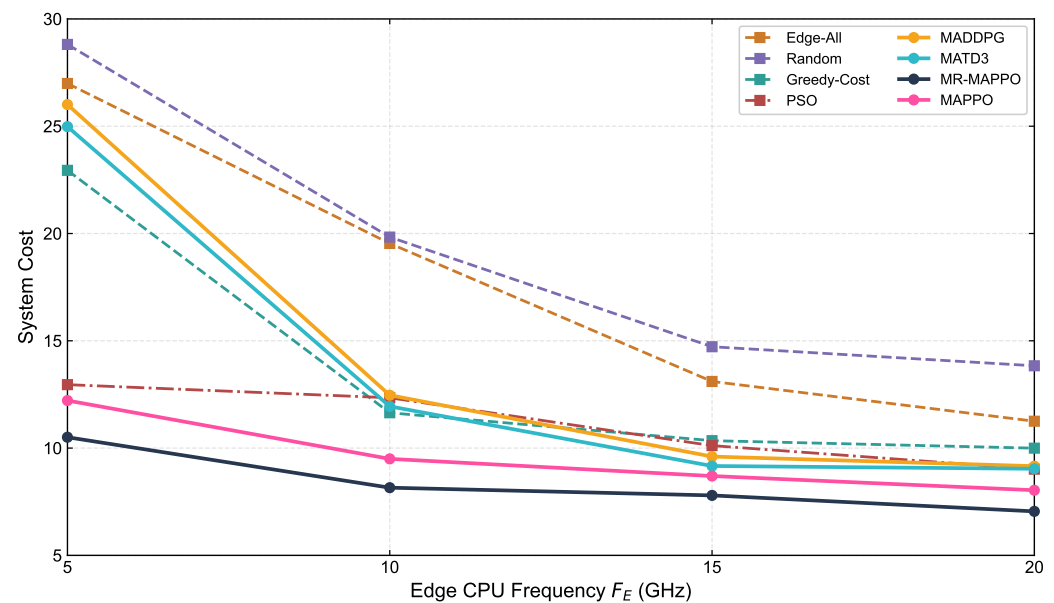


Figure 7. System cost versus edge computing capacity.

Table 3. Runtime and memory overhead of MR-MAPPO with different numbers of UD.

Number of UD N	Training Time s/1000 Steps	Actor-Only Decision Time ms/Slot	Peak Memory MB
20	196.28	0.937	443.95
30	279.33	0.788	444.03
40	335.75	1.089	445.16
50	397.90	0.768	449.43
60	443.89	0.754	450.13

Table 4 reports the system cost under different candidate action sets and BC guidance horizons. Removing the local, edge offloading, or cost-aware candidate increases the system cost by 8.1–10.0%, showing that the three candidate types provide complementary refinement references. Across $M_{BC} = 400, 600$, and 800 , the system cost remains between 7.79 and 8.01, with $M_{BC} = 600$ giving the lowest value. These results indicate that MR-MAPPO is more sensitive to the composition of the candidate action set than to moderate changes in the BC guidance horizon.

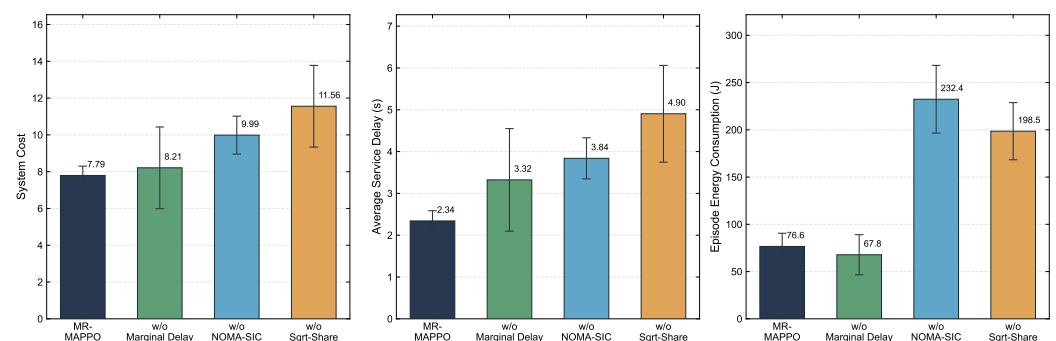
To verify the effectiveness of the communication access mechanism, edge resource scheduling mechanism, and reward design in the proposed MR-MAPPO framework, this paper further designs three key component ablation experiments. The system cost, average service delay, and energy consumption are compared under the same task arrivals, channel states, and edge resource configurations. Specifically, w/o NOMA-SIC removes the uplink NOMA-SIC receiving mechanism, so concurrent offloading users cannot perform successive interference cancellation during co-channel transmission and the system degenerates into a common co-channel interference access model. w/o Sqrt-Share removes the square-root proportional CPU allocation strategy at the ES and replaces multi-task parallel computing scheduling with a first-come-first-served serial processing mechanism. w/o Marginal Delay removes the marginal congestion delay term from the training reward. All ablation configurations are evaluated using the original system cost C_t .

Table 4. Sensitivity of MR-MAPPO to the candidate action set and BC guidance horizon.

Factor	Configuration	System Cost	Relative Increase %
Candidate action set	Full candidate set	7.79	0
	w/o A_t^{loc}	8.57	10.0
	w/o A_t^{edge}	8.42	8.1
	w/o A_t^{cost}	8.47	8.7
BC guidance horizon	$M_{\text{BC}} = 400$	7.84	0.6
	$M_{\text{BC}} = 600$	7.79	0
	$M_{\text{BC}} = 800$	8.01	2.8

Note: Bold entries indicate the configuration adopted in the main experiments.

As shown in Figure 8, removing any of the above components increases the system cost to different degrees. Disabling Sqrt-Share causes a more obvious delay degradation, indicating that concurrent edge computing resource allocation is a key factor in suppressing queue backlogs. Disabling NOMA-SIC leads to a more pronounced increase in energy consumption, showing that increased co-channel interference forces the system to pay higher transmission and scheduling costs. Removing Marginal Delay increases the system cost from 7.79 to 8.21 and the average service delay from 2.34 s to 3.32 s. Although energy consumption decreases, the increased delay results in a higher weighted system cost. This indicates that the marginal congestion delay term helps the policy account for the impact of newly admitted tasks on existing edge workloads and achieve a better delay–energy balance. These results indicate that the performance of MR-MAPPO is not solely attributable to the policy network, but depends on the coordinated effect of the communication access model, the edge concurrent scheduling mechanism, and the congestion-aware reward design.

**Figure 8.** Ablation study of communication, edge scheduling, and reward-design components.

7. Conclusions

This paper studied the joint resource scheduling problem in a single-ES multi-UD dynamic NOMA-MEC system, where offloading admission, communication control, device energy constraints, and concurrent edge workloads were jointly considered under a long-term delay–energy tradeoff objective. The problem involves hybrid discrete–continuous actions, partially observable states, and cross-slot resource coupling, making it difficult to solve directly using static optimization methods. To address this problem, this paper proposed an MR-MAPPO scheduling framework and introduced a master action refinement mechanism during training to improve early-stage policy exploration with physically feasible refined actions. For concurrent edge execution, a closed-form solution to the ES CPU allocation subproblem under a fixed task set was further derived, enabling edge computing resources to be allocated in proportion to the square root of task residual workloads. Simulation results showed that MR-MAPPO outperformed the compared baselines in system cost, training convergence, and scalability with respect to the number of users. Under the main experimental

setting, MR-MAPPO reduced the system cost by 17.9% and 22.9% relative to MAPPO and PSO, respectively. Ablation results showed that the NOMA-SIC access mechanism, the square-root proportional CPU allocation strategy, and the congestion-aware reward design had a significant impact on system performance, indicating that communication access modeling, concurrent edge resource scheduling, and congestion-aware reward design jointly determine the performance of joint scheduling. The current evaluation assumes ideal SIC and perfect CSI and abstracts the ES as a divisible task-level computing resource pool; residual SIC interference and hardware-level CPU scheduling effects are not explicitly modeled. Future work may extend the proposed framework to multi-ES cooperation, user mobility, imperfect SIC conditions, more realistic temporal channel evolution, and hardware-aware edge resource allocation. Robust scheduling under random noise fluctuations, power outages, and UD failures will also be investigated.

Author Contributions: Conceptualization, J.Z.; supervision, J.Z.; review and editing, J.Z.; methodology, S.W.; validation, S.W.; formal analysis, S.W.; investigation, S.W.; writing, S.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Basic Scientific Research Business Fund Project of Higher Education Institutions in Heilongjiang Province, grant number 145409601, and the Natural Science Foundation of Heilongjiang Province, project application number JJ2025XQ0236.

Data Availability Statement: The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: We confirm that AI tools were used only for English language editing and polishing of manuscript, including grammar, wording, and readability improvements. No AI tools were used to generate scientific content, data, figures, analyses, interpretations, or conclusions. The authors carefully reviewed and edited the manuscript and take full responsibility for its content.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AWGN	Additive white Gaussian noise
BC	Behavior cloning
CSI	Channel state information
CTDE	Centralized training with decentralized execution
DDPG	Deep deterministic policy gradient
DRL	Deep reinforcement learning
ES	Edge server
FIFO	First-in-first-out
GAE	Generalized advantage estimation
HOL	Head-of-line
MADDPG	Multi-agent deep deterministic policy gradient
MAPPO	Multi-agent proximal policy optimization
MATD3	Multi-agent twin delayed deep deterministic policy gradient
MEC	Mobile edge computing
MINLP	Mixed-integer nonlinear programming
MR-MAPPO	Master-refined multi-agent proximal policy optimization
MSE	Mean squared error
NOMA	Non-orthogonal multiple access
NOMA-MEC	Non-orthogonal multiple access mobile edge computing
NOMA-SIC	Non-orthogonal multiple access with successive interference cancellation
POMDP	Partially observable Markov decision process

PPO	Proximal policy optimization
SCMA	Sparse code multiple access
SIC	Successive interference cancellation
SINR	Signal-to-interference-plus-noise ratio
TD	Temporal difference
UD	User device

Appendix A. Derivation of the Closed-Form Solution for Edge Computing Resource Allocation

In any decision slot t , when the offloading requests, ES admission decisions, local CPU frequencies, and uplink transmit powers are given, the remaining edge-side resource scheduling problem reduces to allocating ES CPU resources over a fixed edge task set. Given that the considered MEC scenario focuses on energy- and computation-limited UDs and the infrastructure-side ES is not subject to a battery constraint, this subproblem minimizes the sum of edge task computation delays. The corresponding closed-form solution is derived below. The resulting edge computing energy is evaluated in the system cost according to the obtained ES CPU allocation.

The derivation takes the existing edge task set $\mathcal{Q}_t^- \neq \emptyset$ at the beginning of slot t as an example. For any $\iota \in \mathcal{Q}_t^-$, its residual computation workload is $q_{\iota,t}^-$, and the allocated edge CPU frequency is $f_{\iota,t}^-$. According to (22), after fixing the other scheduling variables, the part of problem **P** related to edge CPU allocation can be simplified as

$$\begin{aligned} \mathbf{P}_{\text{edge}}^- : \quad & \min_{\{f_{\iota,t}^-\}} \sum_{\iota \in \mathcal{Q}_t^-} \frac{q_{\iota,t}^-}{f_{\iota,t}^-} \\ \text{s.t.} \quad & \sum_{\iota \in \mathcal{Q}_t^-} f_{\iota,t}^- \leq F_E, \\ & f_{\iota,t}^- > 0, \quad \forall \iota \in \mathcal{Q}_t^-. \end{aligned} \quad (\text{A1})$$

For subproblem $\mathbf{P}_{\text{edge}}^-$, each term $q_{\iota,t}^-/f_{\iota,t}^-$ in the objective function is convex over $f_{\iota,t}^- > 0$, and the constraint is linear. Therefore, $\mathbf{P}_{\text{edge}}^-$ is a convex optimization problem. To avoid calling a numerical optimizer online, this paper derives its closed-form solution using the Lagrange multiplier method. Introduce the Lagrange multiplier $\nu_t \geq 0$ for the edge CPU capacity constraint. The Lagrangian function is constructed as

$$\mathcal{L}(\{f_{\iota,t}^-\}, \nu_t) = \sum_{\iota \in \mathcal{Q}_t^-} \frac{q_{\iota,t}^-}{f_{\iota,t}^-} + \nu_t \left(\sum_{\iota \in \mathcal{Q}_t^-} f_{\iota,t}^- - F_E \right). \quad (\text{A2})$$

To obtain the stationary point, set the first-order partial derivative of $\mathcal{L}$ with respect to the optimization variable $f_{\iota,t}^-$ to zero:

$$\frac{\partial \mathcal{L}}{\partial f_{\iota,t}^-} = -\frac{q_{\iota,t}^-}{(f_{\iota,t}^-)^2} + \nu_t = 0. \quad (\text{A3})$$

This gives the relationship between the edge CPU frequency and the Lagrange multiplier:

$$f_{\iota,t}^- = \sqrt{\frac{q_{\iota,t}^-}{\nu_t}}. \quad (\text{A4})$$

Since the term $q_{\iota,t}^-/f_{\iota,t}^-$ strictly decreases with $f_{\iota,t}^- > 0$, the ES CPU capacity constraint must be active at the optimum:

$$\sum_{l \in \mathcal{Q}_t^-} f_{l,t}^- = F_E. \quad (\text{A5})$$

Substituting $f_{l,t}^- = \sqrt{q_{l,t}^-}/v_t$ into the above capacity constraint yields

$$\sum_{l \in \mathcal{Q}_t^-} \sqrt{\frac{q_{l,t}^-}{v_t}} = F_E. \quad (\text{A6})$$

Further rearrangement gives

$$\frac{1}{\sqrt{v_t}} = \frac{F_E}{\sum_{l \in \mathcal{Q}_t^-} \sqrt{q_{l,t}^-}}. \quad (\text{A7})$$

Finally, substituting this result into $f_{l,t}^- = \sqrt{q_{l,t}^-}/v_t$ gives the closed-form edge computing resource allocation rule:

$$f_{l,t}^- = F_E \frac{\sqrt{q_{l,t}^-}}{\sum_{l \in \mathcal{Q}_t^-} \sqrt{q_{l,t}^-}}, \quad \forall l \in \mathcal{Q}_t^-. \quad (\text{A8})$$

The above result shows that, given the existing edge task set and the objective of minimizing the sum of edge computation delays, the ES computing resources should be allocated in proportion to the square root of each task's residual computation workload. This closed-form solution is used in the main text to compute the progress of existing edge tasks, avoiding repeated online solutions of the edge CPU allocation subproblem in each slot.

Similarly, for the active edge task set $\mathcal{Q}_t^+$ after action execution, replacing $\mathcal{Q}_t^-$ and $q_{l,t}^-$ with $\mathcal{Q}_t^+$ and $q_{l,t}^+$, respectively, gives the edge CPU frequency of the newly admitted task (n, t) :

$$f_{n,t}^{\text{edge}} = F_E \frac{\sqrt{q_{(n,t),t}^+}}{\sum_{l \in \mathcal{Q}_t^+} \sqrt{q_{l,t}^+}}. \quad (\text{A9})$$

References

1. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1628–1656. <https://doi.org/10.1109/COMST.2017.2682318>.
2. Lokavarapu, V.S.; Rao, K.N.; Gokuruboyina, S.; Reddy, S.S. An in-depth analysis of computation offloading approaches in multi-access edge computing systems. *Peer-to-Peer Netw. Appl.* **2025**, *18*, 219. <https://doi.org/10.1007/s12083-025-02028-2>.
3. Qiu, H.; Gao, S.; Chen, Y.; Tu, G. Energy-Efficient Rate Allocation for NOMA-MEC Offloading Under Outage Constraints. *IEEE Commun. Lett.* **2022**, *26*, 2710–2714. <https://doi.org/10.1109/LCOMM.2022.3195767>.
4. Ahn, J.; Mughal, D.M.; Kim, S.H.; Chung, M.Y. Computation rate maximization in active RIS-assisted hybrid FDMA-NOMA MEC systems: A deep reinforcement learning approach. *IEEE Wirel. Commun. Lett.* **2025**, *14*, 1346–1350. <https://doi.org/10.1109/LWC.2025.3542086>.
5. Ren, M.; Chen, J.; Yang, L.; Zhou, Y.; He, B.; Jiang, H. Energy-delay tradeoff in helper-assisted NOMA-MEC systems: A four-sided matching algorithm. *IEEE Trans. Commun.* **2024**, *72*, 2835–2850. <https://doi.org/10.1109/TCOMM.2023.3349166>.
6. Verma, V.R.; Pushkar, B.; Verma, A.; Sharma, V.; Tripathi, P.K. An extensive investigation on Lyapunov optimization-based task offloading techniques in multi-access edge computing. *SN Comput. Sci.* **2025**, *6*, 603. <https://doi.org/10.1007/s42979-025-04130-x>.
7. Wang, J.; Zhang, W.; Lin, Z.; Zheng, Y. Optimal sequential computation offloading and migration in mobile edge computing with multi-task priority. *Computing* **2025**, *107*, 117. <https://doi.org/10.1007/s00607-025-01477-0>.

8. Shang, C.; Sun, Y.; Luo, H.; Guizani, M. Computation Offloading and Resource Allocation in NOMA-MEC: A Deep Reinforcement Learning Approach. *IEEE Internet Things J.* **2023**, *10*, 15464–15476. <https://doi.org/10.1109/JIOT.2023.3264206>.
9. Wang, J.; Wang, Y.; Cheng, P.; Yu, K.; Xiang, W. DDPG-Based Joint Resource Management for Latency Minimization in NOMA-MEC Networks. *IEEE Commun. Lett.* **2023**, *27*, 1814–1818. <https://doi.org/10.1109/LCOMM.2023.3266931>.
10. Gao, Z.; Yang, L.; Dai, Y. Large-Scale Computation Offloading Using a Multi-Agent Reinforcement Learning in Heterogeneous Multi-Access Edge Computing. *IEEE Trans. Mob. Comput.* **2023**, *22*, 3425–3443. <https://doi.org/10.1109/TMC.2022.3141080>.
11. Liu, P.; An, K.; Lei, J.; Sun, Y.; Liu, W.; Chatzinotas, S. Computation Rate Maximization for SCMA-Aided Edge Computing in IoT Networks: A Multi-Agent Reinforcement Learning Approach. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 10414–10429. <https://doi.org/10.1109/TWC.2024.3371791>.
12. Wu, P.; Guan, Y. Multi-agent deep reinforcement learning for computation offloading in cooperative edge network. *J. Intell. Inf. Syst.* **2025**, *63*, 567–591. <https://doi.org/10.1007/s10844-024-00907-3>.
13. Naderializadeh, N.; Sydir, J.; Simsek, M.; Nikopour, H. Resource Management in Wireless Networks via Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*; IEEE: New York, NY, USA, 2020; pp. 1–5. <https://doi.org/10.1109/SPAWC48557.2020.9154250>.
14. Miriyala, S.; Chirra, V.R. Deep learning approaches for computation offloading in edge computing: A critical review. *Telecommun. Syst.* **2026**, *89*, 42. <https://doi.org/10.1007/s11235-026-01426-y>.
15. Zhang, G.; Zhang, W.; Cao, Y.; Li, D.; Wang, L. Energy-Delay Tradeoff for Dynamic Offloading in Mobile-Edge Computing System With Energy Harvesting Devices. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4642–4655. <https://doi.org/10.1109/TII.2018.2843365>.
16. Dash, S.; Mishra, J.; Dash, S.K. Modified Q-Learning Approach for Optimized Task Offloading and Resource Allocation in Mobile Edge Computing. *SN Comput. Sci.* **2026**, *7*, 99. <https://doi.org/10.1007/s42979-025-04673-z>.
17. Aminu, J.; Latip, R.; Hanapi, Z.M.; Kamarudin, S.; Gabi, D.; Shehu, M.A. Weighted quantum particle swarm task offloading optimization algorithm for time-energy minimization in mobile edge computing. *Discov. Comput.* **2026**, *29*, 204. <https://doi.org/10.1007/s10791-026-10087-z>.
18. Lotfolahi, A.; Ferng, H.W. DRL-based resource allocation in NOMA-aided industrial IoT towards energy productivity maximization. *IEEE Trans. Netw. Sci. Eng.* **2026**, *13*, 469–484. <https://doi.org/10.1109/TNSE.2025.3584786>.
19. Lu, B.; Fang, J.; Hong, X.; Shi, J. Task offloading in dynamic energy splitting STAR-RIS assisted NOMA-MEC systems with decomposition based multi-agent DRL. *IEEE Trans. Veh. Technol.* **2025**, *74*, 13091–13103. <https://doi.org/10.1109/TVT.2025.3557279>.
20. Wu, G.; Chen, G.; Gu, X. Multi-agent learning based UAV-assisted MEC-NOMA for cellular offloading in Internet of Things. *Expert Syst. Appl.* **2025**, *270*, 126567. <https://doi.org/10.1016/j.eswa.2025.126567>.
21. Zhang, L.; Lu, X.; Liu, J.; Zhang, Y.; Chen, J.; Cao, B.; Li, K. Multi-agent reinforcement learning for resource allocation in NOMA-enhanced aerial edge computing networks. *J. Syst. Archit.* **2026**, *170*, 103634. <https://doi.org/10.1016/j.sysarc.2025.103634>.
22. Zhou, Z.; Zhang, Z.; Zeng, J.J.; Chao, H.C. Multi-Agent Transformer approach for collaborative task offloading and resource optimization in NOMA-based Vehicular Edge Computing. *Ad Hoc Netw.* **2026**, *187*, 104222. <https://doi.org/10.1016/j.adhoc.2026.104222>.
23. Zhang, G.; Song, J.; Hu, Y.; Fan, P.; Li, C.; Zhou, X. Distributed Task Scheduling Algorithm Based on MADDPG in MEC Emergency Networks. *IEEE Internet Things J.* **2025**, *12*, 44692–44704. <https://doi.org/10.1109/JIOT.2025.3581514>.
24. Pan, Z.; Jiang, Y.; Hang, Y.; Zheng, F.C. Energy-Optimized Computation Offloading in MEC-Enabled Cell-Free Massive MIMO Systems with MADDPG. In *Proceedings of the 2025 IEEE 102nd Vehicular Technology Conference (VTC2025-Fall)*; IEEE: New York, NY, USA, 2025; pp. 1–6. <https://doi.org/10.1109/VTC2025-Fall65116.2025.11310087>.
25. Lai, J.; Liu, H.; Xu, G.; Jiang, W.; Wang, X.; Jiang, D. Joint computation offloading and resource allocation for LEO satellite networks using hierarchical multi-agent reinforcement learning. *IEEE Trans. Cogn. Commun. Netw.* **2024**, *11*, 2554–2567. <https://doi.org/10.1109/TCCN.2024.3510562>.
26. Long, S.; Rao, C.; Liu, H.; Chen, Y.; Deng, Q.; Shang, J.; Li, Z. Fault-Tolerant Aware Task Offloading Based on Reinforcement Learning in Mobile Edge Computing. *IEEE Trans. Mob. Comput.* **2026**, *25*, 6068–6082. <https://doi.org/10.1109/TMC.2025.3636100>.
27. Ghahari-Bidgoli, M.; Arani, M.G.; Sharif, A. JORAS: Joint offloading and resource auto-scaling in edge computing environment. *Clust. Comput.* **2025**, *28*, 1066. <https://doi.org/10.1007/s10586-025-05756-5>.
28. Zhang, Z.; Xu, C.; Ma, L.; Cai, Y.; Lin, F.; Xie, X. Dual-Mode Energy Harvesting-Based Aerial Computing in NOMA-IoV Networks via Deep Reinforcement Learning. *IEEE Internet Things J.* **2026**, *13*, 6294–6308. <https://doi.org/10.1109/JIOT.2025.3635213>.
29. Lyu, S.; Feng, M.; Xiao, L.; Zhou, J.; Jiang, T. Intelligent Task Offloading and Resource Allocation for NOMA-based Multi-beam Satellite Mobile Edge Computing Systems. *IEEE Trans. Wirel. Commun.* **2026**, *25*, 4143–4155. <https://doi.org/10.1109/TWC.2025.3608832>.
30. Liang, M.; Wang, B. A Hybrid Heuristic Task Offloading Algorithm in Mobile Edge Computing for Joint Optimization of Delay and Energy Consumption. *J. Netw. Syst. Manag.* **2025**, *33*, 65. <https://doi.org/10.1007/s10922-025-09945-w>.

31. Chahoud, M.; Sami, H.; Mizouni, R.; Bentahar, J.; Mourad, A.; Otrouk, H.; Talhi, C. Reward Shaping in DRL: A Novel Framework for Adaptive Resource Management in Dynamic Environments. *Inf. Sci.* **2025**, *715*, 122238. <https://doi.org/10.1016/j.ins.2025.122238>.
32. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. <https://doi.org/10.48550/arXiv.1707.06347>.
33. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.I.; Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv* **2015**, arXiv:1506.02438. <https://doi.org/10.48550/arXiv.1506.02438>.
34. Qin, L.; Lu, H.; Chen, Y.; Chong, B.; Wu, F. Toward decentralized task offloading and resource allocation in user-centric MEC. *IEEE Trans. Mob. Comput.* **2024**, *23*, 11807–11823. <https://doi.org/10.1109/TMC.2024.3399766>.
35. Zhou, J.; Hou, X.; Zeng, Y.; Cong, P.; Jiang, W.; Guo, S. Quality of Experience and Reliability-Aware Task Offloading and Scheduling for Multi-User Mobile-Edge Computing Systems. *IEEE Trans. Serv. Comput.* **2025**, *18*, 1683–1696. <https://doi.org/10.1109/TSC.2025.3552338>.
36. Li, M.; Feng, Z.; Li, B. Efficient joint optimization of task offloading and resource allocation for LEO satellite edge computing. *Ad Hoc Netw.* **2026**, *182*, 104097. <https://doi.org/10.1016/j.adhoc.2025.104097>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.