Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.084892

ARTICLE

Cooperative Task Offloading in Mobile Edge Computing via an Improved MASAC Framework

Zheng Yao[1], Jie Liu[1], Changjun Deng[2,3,*] and Wang Lin[2,3]

[1]School of Robotics and Automation, Hubei University of Automotive Technology, Shiyan, China
[2]Yunnan International Joint Laboratory of Agricultural Remote Sensing and Digital Technology, Yunnan Hanzhe Technology Co., Ltd., Kunming, China
[3]Key Laboratory of Plateau Agricultural Environment Monitoring and Control, Ministry of Agriculture and Rural Affairs, Kunming, China

*Corresponding Author: Changjun Deng. Email: hanzhekj@163.com

Received: 06 May 2026; Accepted: 30 June 2026; Published: 13 August 2026

ABSTRACT: Mobile edge computing (MEC) is an effective paradigm for supporting latency-sensitive and computation-intensive intelligent applications. However, in dynamic mobile-edge network scenarios, mobile terminals experience time-varying wireless links due to mobility. Tasks may also arrive unpredictably, while multiple terminals compete for limited edge resources. As a result, MEC systems may suffer from service congestion and unbalanced resource utilization, which increases end-to-end latency and energy consumption. This paper investigates cooperative task offloading in dynamic MEC networks. The considered system comprises one macro base station and multiple small base stations equipped with edge-computing resources. In each time slot, each mobile terminal selects a service option, determines the task offloading ratio, and chooses its transmit power for task uploading. This sequential decision process is formulated as a multi-agent problem with continuous action spaces. Under the centralized training and decentralized execution (CTDE) framework, the problem is further modeled as a decentralized partially observable Markov decision process (Dec-POMDP). Standard multi-agent soft actor-critic (MASAC) is not fully suitable for this problem. Its original action model does not handle bounded continuous actions well. Its exploration strength may also be unsuitable at different training stages. Frequent policy updates can further make training unstable when critic estimates are inaccurate. To address these issues, this paper develops an adaptive Beta-policy and delayed-update multi-agent soft actor-critic method, abbreviated as ABDMASAC. This method uses a Beta policy to model bounded actions. It adjusts the entropy coefficient during training and delays policy updates to reduce training oscillations. Experimental results show that, under a unified training budget and a consistent evaluation protocol, the proposed method achieves a better overall trade-off than the selected MASAC-backbone and on-policy MARL baselines under the considered simulation settings in terms of overall reward, average end-to-end latency, and average energy consumption. In the large-scale scenario, compared with MASAC, it improves the overall reward by 17.8%, reduces the average end-to-end latency by 18.0%, and lowers the average energy consumption by 11.4%.

KEYWORDS: Mobile edge computing; multi-agent reinforcement learning; MASAC; cooperative task offloading

1 Introduction

Intelligent mobile applications such as environmental perception, object recognition, path planning, and cooperative control require timely computation, yet mobile terminals are limited by onboard computing

capacity and battery energy. Cloud-only execution provides abundant resources but introduces long transmission paths and response delay. MEC mitigates this limitation by placing computation near terminals and enabling task offloading between local and edge resources.

Cooperative offloading remains difficult in dynamic MEC networks because terminal mobility changes wireless links, stochastic arrivals create time-varying workloads, and multiple terminals compete for shared edge resources. This paper therefore studies cooperative task offloading in dynamic MEC networks, jointly considering node association, task offloading ratio, and transmit-power control under communication, computation, and queueing constraints. The resulting problem is modeled as a multi-agent continuous decision-making problem under CTDE, and an improved MASAC-based method is developed to address bounded action representation, exploration adaptation, and training stability. The main contributions are summarized as follows.

(1) An MEC system model is established for cooperative task offloading in dynamic mobile-edge network scenarios. The model captures terminal mobility, time-varying wireless links, spatially heterogeneous task arrivals, and shared edge-resource competition. Based on this model, the task-offloading problem is formulated to reduce latency and energy consumption while suppressing queue accumulation.
(2) Under the CTDE framework, node association, task offloading, and transmit power control are unified into a multi-agent joint decision-making problem. Considering local observation constraints and joint state transition characteristics, the problem is further modeled as a decentralized partially observable Markov decision process (Dec-POMDP).
(3) To adapt MASAC to the bounded and relaxed hybrid action structure of cooperative MEC offloading, an improved MASAC-based method, named ABDMASAC, is developed. Different from standard MASAC, the proposed method jointly incorporates Beta-policy-based bounded action modeling, adaptive entropy regulation, and delayed actor updates into the CTDE framework. This design enables the agents to learn node-association preferences, offloading ratios, and transmit-power decisions under dynamic link states, stochastic task arrivals, and shared edge-resource competition. These modules are not claimed to independently improve all performance metrics. Instead, they are designed as an integrated learning-stabilization mechanism, where the Beta policy provides bounded action representation, adaptive entropy tuning regulates exploration, and delayed actor updates improve late-stage training stability.
(4) Comparative experiments are conducted under a unified training budget and a consistent evaluation protocol. The results show that ABDMASAC achieves a better overall trade-off than the selected MASAC-backbone and on-policy MARL baselines in terms of system reward, end-to-end latency, and terminal-side energy consumption.

The remainder of this paper is organized as follows. Section 2 gives the system model and problem formulation. Section 3 presents ABDMASAC. Section 4 reports experiments and ablations, and Section 5 concludes the paper.

1.1 Related Work

1.1.1 Optimization-Based MEC Task Offloading

Optimization-based MEC studies have modeled offloading through power-delay tradeoffs, resource allocation, trajectory planning, dense-user interaction, task heterogeneity, and multi-objective scheduling. Representative works include Chen et al. [1], Mao et al. [2], Wu et al. [3], Yang et al. [4], Li et al. [5], Yao et al. [6], Gao et al. [7], Tong et al. [8], Han et al. [9], Misha et al. [10], and Yao and Chang [11]. Surveys by Mao et al. [12], Feng et al. [13], Kar et al. [14], Dong et al. [15], and Wang et al. [16] show that MEC offloading

has evolved toward joint latency-energy-resource optimization, but many models still rely on simplified dynamics and coordination assumptions.

1.1.2 DRL-Based MEC Task Offloading

DRL has been used to improve adaptability when MEC states and constraints are difficult to optimize analytically. Surveys by Luo and Dai [17], Hortelano et al. [18], Zabihi et al. [19], Peng et al. [20], and Nabi et al. [21] summarize this trend. Representative studies include Tang and Wong [22], Mirza et al. [23], Liu et al. [24], Liu et al. [25], and Xie et al. [26]. More recently, Mustafa et al. [27] developed a Dueling-DQN-based method for partial computation offloading and resource allocation in MEC, further demonstrating the effectiveness of value-based DRL in jointly optimizing task partitioning and resource-control decisions. However, single-agent DRL is less suitable when multiple terminals jointly affect node load, delay, energy, and queue evolution.

1.1.3 MARL-Based Cooperative Offloading

MARL-based MEC offloading has considered communication-assisted coordination, UAV trajectory coupling, prioritized replay, and hybrid action spaces, as shown by Tan et al. [28], Ju et al. [29], Shi et al. [30], and Wang et al. [31]. The evolutionary-game-and-MATD3-based joint node-selection and task-offloading method of Yao et al. [32] further addresses UAV-assisted MEC with hybrid discrete-continuous action spaces. Chen et al. [33], Zuo et al. [34], Li et al. [35], and Bo and Zhao [36] extend MARL offloading to LEO, SAGIN, V2X, and vehicle-edge settings. These studies support cooperative learning, but bounded hybrid actions and stable CTDE training remain challenging under mobility, stochastic arrivals, and edge-resource contention.

1.1.4 Research Gap and Motivation

Existing studies usually emphasize discrete association, continuous resource control, or simplified coordination. The joint treatment of relaxed service-node association, bounded task partitioning and power control, time-varying links, stochastic arrivals, and shared edge-resource competition remains insufficient, motivating the proposed ABDMASAC framework under CTDE.

2 System Model and Problem Formulation

This section defines the dynamic MEC model and the cooperative offloading problem. Robot terminals in Fig. 1 are only an example; the formulation uses the general term “terminal”.

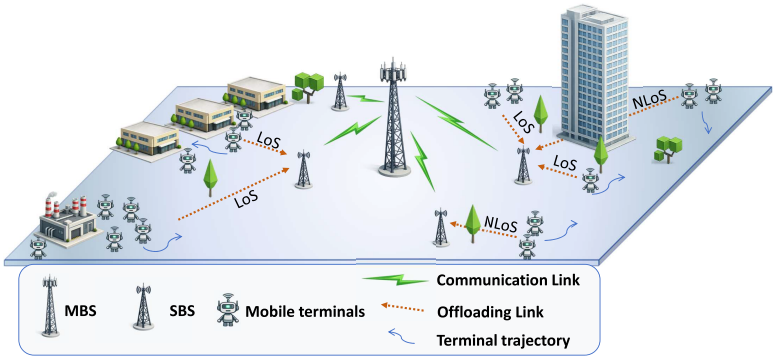

Figure 1: A representative mobile-terminal MEC scenario for cooperative task offloading.

2.1 System Model

Consider the heterogeneous MEC system in Fig. 1, with multiple mobile terminals, one MBS, and K SBSs. The MBS and SBSs are edge nodes, and each task can be processed locally, by an SBS, or by the MBS.

Let the terminal set be defined as $\mathcal{N} = \{1, 2, \ldots, N\}$, where N denotes the number of mobile terminals. Let K denote the number of SBSs and $J = K + 1$ denote the number of edge nodes, including one MBS and K SBSs. The service-option set is defined as $\mathcal{S} = \{0, 1, \ldots, J\}$, where 0 denotes local execution, 1 denotes the MBS, and $2, \ldots, J$ denote the candidate SBSs. The edge-node set is defined as $\mathcal{B} = \mathcal{S}\backslash\{0\}$.

The system evolves in region $\mathcal{R} \subset \mathbb{R}^2$ over slots $\mathcal{T} = \{0, 1, \ldots, T-1\}$ of length Δt. Each terminal selects a service option, offloading ratio, and transmit power from local observations, while edge resources are shared according to node load.

Let the two-dimensional positions of terminal n and service nodes be defined in Eq. (1):

$$\begin{aligned} p_n(t) &= [x_n(t), y_n(t)]^T, \quad n \in \mathcal{N},\ t \in \mathcal{T}, \\ q_1 &= [x_1^{\mathrm{mbs}}, y_1^{\mathrm{mbs}}]^T, \\ q_j &= [x_j^{\mathrm{sbs}}, y_j^{\mathrm{sbs}}]^T, \qquad j \in \{2, \ldots, J\}. \end{aligned} \tag{1}$$

Terminal mobility is modeled by the Gaussian random walk in Eq. (2):

$$p_n(t+1) = \Pi_{\mathcal{R}}(p_n(t) + \Delta p_n(t)), \tag{2}$$

where $\Pi_{\mathcal{R}}(\cdot)$ keeps the terminal inside $\mathcal{R}$, and $\Delta p_n(t) \sim N(0, \sigma_m^2 I)$.

The selected service option is $j_n(t) \in \mathcal{S}$, where 0, 1, and $2, \ldots, J$ denote local execution, the MBS, and SBSs, respectively. Positions, selections, and arrivals jointly determine node load.

2.2 Communication Model

Let $c_{n,j}(t) \in \{0, 1\}$ denote the NLoS/LoS state of link (n, j). Temporal blockage correlation is described by a two-state Markov chain $\Pr(c_{n,j}(t+1) \mid c_{n,j}(t))$.

Let the Euclidean distance between terminal n and edge node $j \in \mathcal{B}$ at slot t and the corresponding channel gain be defined by Eq. (3):

$$\begin{aligned} d_{n,j}(t) &= \|p_n(t) - q_j\|_2, \\ h_{n,j}(t) &= h(d_{n,j}(t), c_{n,j}(t)). \end{aligned} \tag{3}$$

Here, $\|\cdot\|_2$ is the Euclidean norm, and $h(\cdot)$ depends on distance and link state.

Let $u_n(t) \in [0, 1]$ denote the normalized transmit-power control variable of terminal n at slot t. The corresponding actual transmit power is defined as $P_n(t) = u_n(t) P_{\max}$, where $P_{\max}$ denotes the maximum transmit power.

An effective orthogonal uplink access model is used: concurrent uploads occupy orthogonal resource blocks or an effective allocated bandwidth, so co-channel inter-user interference is not explicitly modeled. Thus, B in Eq. (4) is the effective uplink bandwidth.

The uplink transmission rate from terminal n to edge node $j \in \mathcal{B}$ at slot t is given in Eq. (4):

$$R_{n,j}(t) = B \log_2 \left(1 + \frac{x_{n,j}(t) P_n(t) h_{n,j}(t)}{\sigma^2}\right), \tag{4}$$

where B denotes the effective allocated uplink bandwidth under the orthogonal-access assumption, $x_{n,j}(t)$ indicates whether edge node j is selected for task transmission, and σ^2 denotes the noise power.

Accordingly, if the amount of offloaded task data of terminal n at slot t is $D_n^{off}(t)$, the uplink transmission delay and the corresponding transmission energy consumption to edge node $j \in \mathcal{B}$ are defined in Eq. (5):

$$\begin{aligned} T_{n,j}^{tx}(t) &= \frac{D_n^{off}(t)}{R_{n,j}(t)}, \\ E_{n,j}^{tx}(t) &= P_n(t) T_{n,j}^{tx}(t). \end{aligned} \tag{5}$$

2.3 Computation Model

Assume that the task generated by terminal n at slot t is represented by the triplet in Eq. (6):

$$\Omega_n(t) = (D_n(t), C_n(t), \tau_n^{\max}(t)). \tag{6}$$

Here, $D_n(t)$, $C_n(t)$, and $\tau_n^{\max}(t)$ denote data size, CPU cycles per bit, and latency tolerance, respectively.

Let $Q_n(t)$ be the terminal-side task queue. Its evolution is

$$Q_n(t+1) = \max\{Q_n(t) - S_n(t), 0\} + A_n(t), \tag{7}$$

where $A_n(t)$ and $S_n(t)$ denote newly arrived and completed tasks.

With offloading ratio $\rho_n(t) \in [0,1]$, local and offloaded data are

$$D_n^{loc}(t) = (1 - \rho_n(t)) D_n(t), D_n^{off}(t) = \rho_n(t) D_n(t). \tag{8}$$

For local frequency $f_n^{loc}(t)$, local delay and energy are

$$T_n^{loc}(t) = \frac{D_n^{loc}(t) C_n(t)}{f_n^{loc}(t)}, E_n^{loc}(t) = \kappa (f_n^{loc}(t))^2 D_n^{loc}(t) C_n(t), \tag{9}$$

where κ is the effective capacitance coefficient.

For offloaded data, let F_j be the capacity of node j and $\mathcal{N}_j(t) = \{n \in \mathcal{N} \mid \bar{j}_n(t) = j\}$. The edge-processing delay is $T_{n,j}^{edge}(t) = \frac{D_n^{off}(t) C_n(t)}{f_{n,j}(t)}$. Under equal-share CPU allocation, $f_{n,j}(t) = \frac{F_j}{|\mathcal{N}_j(t)|}$ for $n \in \mathcal{N}_j(t)$.

This equal-share rule captures edge-resource contention. The model does not introduce a persistent server-side queue or explicit edge waiting time; backlog is tracked at the terminal side through $Q_n(t)$.

2.4 Problem Formulation

At slot t, terminal n controls association $x_{n,j}(t)$, offloading ratio $\rho_n(t) \in [0,1]$, and normalized power $u_n(t) \in [0,1]$. The edge CPU share $f_{n,j}(t)$ is induced by node selection and load, not treated as an independent action.

The mixed decisions are relaxed into the continuous action vector

$$a_n(t) = (g_n(t), \rho_n(t), u_n(t)). \tag{10}$$

The node-association gating vector is defined in Eq. (11):

$$g_n(t) = [g_{n,j}(t)]_{j \in \mathcal{S}}. \tag{11}$$

Here, $g_{n,j}(t)$ is the association preference for option j. The executable option is obtained by

$$\hat{j}_n(t) = \arg\max_{j \in \mathcal{S}} g_{n,j}(t). \tag{12}$$

Accordingly, the binary association indicator is defined in Eq. (13):

$$x_{n,j}(t) = \begin{cases} 1, & j = \hat{j}_n(t), \\ 0, & \text{otherwise}, \end{cases} \qquad j \in \mathcal{S}. \tag{13}$$

The gating vector is a continuous relaxation of discrete association. The argmax projection in Eq. (12) is used only during environment interaction and is not treated as differentiable. The policy is optimized in the relaxed continuous space, while the projected hard association is used to compute delay, energy, and queue evolution; hence the method is a relaxation-based hybrid-action policy learner.

Accordingly, the total end-to-end latency of terminal n at slot t is defined in Eq. (14):

$$T_n(t) = \max\left(T_n^{\text{loc}}(t), \sum_{j \in \mathcal{B}} x_{n,j}(t) \left(T_{n,j}^{\text{tx}}(t) + T_{n,j}^{\text{edge}}(t) \right) \right), \tag{14}$$

and the corresponding terminal-side energy consumption is defined in Eq. (15):

$$E_n(t) = E_n^{\text{loc}}(t) + \sum_{j \in \mathcal{B}} x_{n,j}(t)\, E_{n,j}^{\text{tx}}(t). \tag{15}$$

The per-slot cost combines latency, energy, queue length, and overload:

$$c_n(t) = w_d\, \bar{d}_n(t) + w_e\, \bar{e}_n(t) + w_q\, \bar{q}_n(t) + w_o\, \bar{o}_n(t), \tag{16}$$

where w_d, w_e, w_q, and w_o denote the weights associated with latency, energy consumption, queue length, and overload penalty, respectively.

The normalized terms are

$$\bar{d}_n(t) = \frac{T_n(t)}{d_{\text{ref}}}, \quad \bar{e}_n(t) = \frac{E_n(t)}{e_{\text{ref}}}, \quad \bar{q}_n(t) = \frac{Q_n(t)}{Q_{\text{ref}}}, \quad \bar{o}_n(t) = \frac{\max\left(Q_n(t) - Q_{\text{th}}, 0\right)}{Q_{\text{ref}}}, \tag{17}$$

where d_{ref}, e_{ref}, and Q_{ref} denote the reference scales for latency, energy consumption, and queue length, respectively, and Q_{th} denotes the queue-overload threshold.

At the system level, the average cost at slot t is defined in Eq. (18):

$$\bar{C}(t) = \frac{1}{N} \sum_{n=1}^{N} c_n(t). \tag{18}$$

Accordingly, the long-term cooperative offloading objective can be formulated as Eq. (19):

$$\min\, \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t\, \bar{C}(t) \right]. \tag{19}$$

where $\gamma \in (0,1)$ is the discount factor.

The resulting problem is

$$
\begin{aligned}
\mathbf{P1}: \quad & \min_{\{x_{n,j}(t),\rho_n(t),u_n(t)\}} \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t \bar{C}(t)\right] \\
\text{s.t.} \quad & x_{n,j}(t) \in \{0,1\}, \quad \forall n \in \mathcal{N},\ j \in \mathcal{S},\ t \in \mathcal{T}, \\
& \sum_{j \in \mathcal{S}} x_{n,j}(t) = 1, \quad \forall n \in \mathcal{N},\ t \in \mathcal{T}, \\
& 0 \le \rho_n(t) \le 1, \quad 0 \le u_n(t) \le 1, \quad \forall n \in \mathcal{N},\ t \in \mathcal{T}, \\
& Q_n(t+1) = \max\{Q_n(t) - S_n(t), 0\} + A_n(t), \\
& f_{n,j}(t) = \frac{F_j}{|\mathcal{N}_j(t)|}, \quad n \in \mathcal{N}_j(t),\ j \in \mathcal{B}.
\end{aligned}
\tag{20}
$$

The constraints specify binary one-option association, bounded continuous controls, queue evolution, and equal-share CPU allocation.

Because P1 couples binary association, bounded continuous control, stochastic transitions, and multiple agents, it is solved as a Dec-POMDP under CTDE.

The shared reward is the negative average cost:

$$r(t) = -\bar{C}(t). \tag{21}$$

Reward clipping is used for critic stability:

$$\tilde{r}(t) = \mathrm{clip}(r(t), -10, 0). \tag{22}$$

3 Cooperative Offloading Optimization Method Based on Improved MASAC

Following CTDE [37], P1 is formulated as a Dec-POMDP and solved by the proposed ABD-MASAC algorithm.

3.1 Dec-POMDP Formulation

Each terminal is an agent, and the agent set is

$$\mathcal{N} = \{1, 2, \ldots, N\}. \tag{23}$$

The global state $s(t) \in X$ contains terminal positions, link states, queues, and node loads. Agent n observes $o_n(t) \in O_n$, including its position, task state, reachable-link information, and local resources. The joint observation is

$$o(t) = (o_1(t), o_2(t), \ldots, o_N(t)). \tag{24}$$

The local action $a_n(t) \in A_n$ follows Eq. (10): the gating vector represents relaxed node association, $\rho_n(t)$ is the offloading ratio, and $u_n(t)$ is normalized transmit power. The discrete service option is obtained by Eq. (12). The joint action is

$$a(t) = (a_1(t), a_2(t), \ldots, a_N(t)). \tag{25}$$

The transition probability is

$$\mathbb{P}(s(t+1) \mid s(t), a(t)), \tag{26}$$

which is driven by mobility, link switching, task arrivals, and shared edge-resource competition. The shared reward follows Eq. (21); maximizing its discounted return is equivalent to minimizing the cost in P1.

3.2 Improved Learning Framework Based on MASAC

ABDMASAC builds on SAC [38] and CTDE [37], and improves MASAC through bounded action modeling, adaptive entropy tuning, and delayed policy updates.

3.2.1 CTDE Learning Framework

Under CTDE, each terminal executes its actor using only local observations, while centralized critics use global information and joint actions for training. Fig. 2 shows the framework.

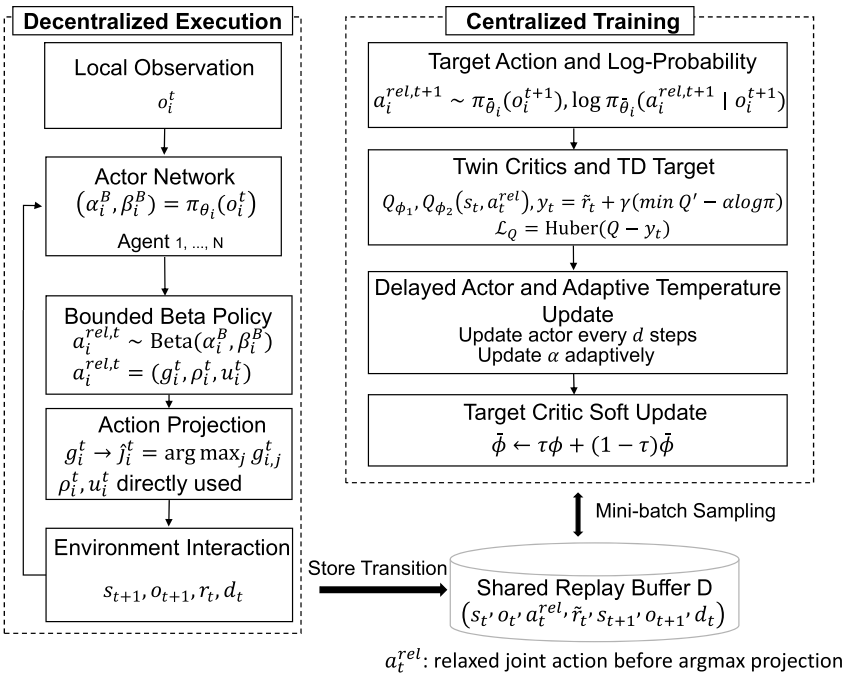

Figure 2: Overall framework of the proposed ABDMASAC under centralized training and decentralized execution. The replay buffer stores the relaxed continuous joint action a_t^{rel}, while the hard service-option selection is obtained by argmax projection only during environment interaction.

Let the actor-network parameters of terminal n be denoted by θ_n. The actor takes the local observation o_t^n as input and outputs the continuous action a_t^n. Twin critics $Q_{\phi_n^j}(s_t, a_t)$, $j \in \{1, 2\}$, use the global state and relaxed joint action; target networks and Huber loss improve stability [38,39]. The critic loss is

$$L(\phi_n^j) = E_{(s_t, a_t, r_t, s_{t+1}) \sim D}\left[\mathcal{H}_\delta\left(Q_{\phi_n^j}(s_t, a_t) - y_t\right)\right], j \in \{1, 2\}, \tag{27}$$

where D denotes the replay buffer, $\mathcal{H}_\delta(\cdot)$ denotes the Huber loss, and the target value y_t is defined in Eq. (28):

The replay buffer stores the relaxed continuous action vector, including gating, offloading ratio, and normalized power. Argmax-based hard association is used only by the environment, whereas the centralized critic uses the global state and relaxed joint action for value gradients.

$$y_t = r_t + \gamma \left(\min_{k \in \{1,2\}} Q_{\phi_n^{k'}}(s_{t+1}, a_{t+1}) - \alpha \log \pi_{\theta_n}(a_{t+1}^n \mid o_{t+1}^n) \right) \tag{28}$$

3.2.2 Beta Bounded Policy

Because gating variables, offloading ratio, and normalized power are bounded, ABDMASAC follows Chou et al. [40] and models each continuous action dimension with a Beta distribution.

The actor outputs $(\alpha_t^n, \beta_t^n) = f_{\theta_n}(o_t^n)$, and samples

$$a_t^n \sim \text{Beta}(\alpha_t^n, \beta_t^n). \tag{29}$$

In the implementation, the actor network outputs two unconstrained vectors, which are transformed into positive Beta concentration parameters through the softplus function. To avoid numerical instability near degenerate Beta distributions, the concentration parameters are clipped to $[10^{-3}, 100]$. During training, stochastic actions are sampled using the reparameterized sampling function of the Beta distribution, which provides a pathwise gradient estimator for the continuous relaxed action. During deterministic evaluation, the mean of the Beta distribution is used. The log-probability of the sampled bounded action is used in the entropy-regularized actor objective.

For $x \in (0,1)$, the Beta density is

$$\pi(x; \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, x \in (0,1), \tag{30}$$

where $\Gamma(\cdot)$ is the Gamma function. At execution, gating is projected by Eq. (12), while $\rho_n(t)$ and $u_n(t)$ remain continuous controls.

The entropy-regularized actor objective is

$$J(\theta_n) = E_{(s_t, a_t) \sim D} \left[\alpha \log \pi_{\theta_n}(a_t^n \mid o_t^n) - Q_{\phi_n}(s_t, a_t) \right] \tag{31}$$

3.2.3 Adaptive Entropy Tuning

ABDMASAC uses adaptive temperature α [38] to regulate exploration across training stages.

The temperature objective is

$$J(\alpha) = E_{a_t^n \sim \pi_{\theta_n}} \left[-\alpha (\log \pi_{\theta_n}(a_t^n \mid o_t^n) + \bar{H}) \right], \tag{32}$$

where $\bar{H}$ is the target entropy. Minimizing $J(\alpha)$ adjusts exploration according to the gap between current and target entropy.

3.2.4 Delayed Policy Update

To reduce actor oscillation caused by transient critic errors, ABDMASAC adopts delayed policy updates inspired by Fujimoto et al. [39].

The actor is updated only every d critic updates:

$$\theta_n \leftarrow \theta_n - \eta_\theta \nabla_{\theta_n} J(\theta_n), \quad \text{if } t \bmod d = 0, \tag{33}$$

here, d is the delay interval and η_θ is the actor learning rate. Target critics use soft updates $\phi_n^{j'} \leftarrow \tau\phi_n^j + (1-\tau)\phi_n^{j'}$, $j \in \{1,2\}$.

3.3 Network Update and Training Procedure

During interaction, each actor samples a bounded relaxed action from its Beta policy; the gating part is projected to a service option only for environment execution. The environment returns $r(t)$ and $s(t+1)$, and $\tilde{r}(t) = \mathrm{clip}(r(t), -10, 0)$ is stored.

The interaction sample is stored in the replay buffer as $(s(t), o(t), a^{\mathrm{rel}}(t), \tilde{r}(t), s(t+1), o(t+1), done(t))$, where $a^{\mathrm{rel}}(t)$ denotes the relaxed continuous joint action before the argmax projection, and $done(t)$ denotes the terminal flag.

Unless specified, a_t in learning objectives denotes the relaxed continuous joint action.

Mini-batches update the twin critics using soft Bellman targets and Huber loss. The learned value estimates the long-term cooperative objective containing latency, energy, and queue terms.

Actors and entropy coefficient α are updated according to the delay interval and target-entropy objective.

Algorithm 1 summarizes the complete training procedure of ABDMASAC under the CTDE framework.

Algorithm 1: Training procedure of ABDMASAC under the CTDE framework

Require: Agent set $\mathcal{N}$, total training steps T, replay buffer D, mini-batch size B, discount factor γ, soft-update coefficient τ, policy-delay interval d, and target entropy $\bar{H}$

Ensure: Decentralized actor policies $\{\pi_{\theta_i}\}_{i\in\mathcal{N}}$ and centralized twin critics

Initialize the environment, actors, twin critics, target critics, entropy coefficient α, and replay buffer $\mathcal{D}$
Obtain initial global state s_0 and local observations $o_0 = \{o_i(0)\}_{i\in\mathcal{N}}$
for $t = 0, 1, \ldots, T-1$ **do**
  for each agent $i \in \mathcal{N}$ **do**
    Sample relaxed bounded action $a_i^{\mathrm{rel}}(t) = [g_i(t), \rho_i(t), u_i(t)] \sim \pi_{\theta_i}(\cdot \mid o_i(t))$
    Apply argmax projection to $g_i(t)$ for executable service selection; use $\rho_i(t)$ and $u_i(t)$ as continuous controls
  end for
  Execute the projected joint action and observe s_{t+1}, o_{t+1}, reward r_t, and terminal flag d_t
  Clip reward as $r_t^{\mathrm{clip}} = \mathrm{clip}(r_t, r_{\min}, r_{\max})$
  Store $(s_t, o_t, a_t^{\mathrm{rel}}, r_t^{\mathrm{clip}}, s_{t+1}, o_{t+1}, d_t)$ in $\mathcal{D}$, where a_t^{rel} is before argmax projection
  if $|\mathcal{D}|$ is sufficient for updating **then**
    Sample a mini-batch from $\mathcal{D}$
    Generate next-step relaxed actions using target actors and compute their log probabilities
    Construct entropy-regularized soft Bellman targets using target twin critics
    Update twin critics by minimizing the Huber loss
    if $t \bmod d = 0$ **then**
      Update actors using the entropy-regularized objective
      Update entropy coefficient α toward target entropy $\bar{H}$
      Soft-update target critic networks with coefficient τ

(Continued)

Algorithm 1 (continued)

20: **end if**
21: **end if**; Set $s_t \leftarrow s_{t+1}$ and $o_t \leftarrow o_{t+1}$
22: **end for**

4 Simulation Results and Analysis

This section compares ABDMASAC with MASAC and MAPPO in small- and large-scale MEC scenarios and reports an ablation study.

4.1 Simulation Settings

The testbed contains one MBS and multiple SBSs. The small-scale scenario uses $N = 5$ terminals and $K = 3$ SBSs, while the large-scale scenario uses $N = 20$ and $K = 4$. Terminals follow Gaussian mobility, links follow LoS/NLoS transitions, and hotspot arrivals plus equal-share CPU allocation create controlled resource contention. Parameters are listed in Table 1; richer mobility, interference, bandwidth, CPU scheduling, and edge-server queueing are left for future work.

Table 1: Environment and scenario parameters.

Parameter	Value
Number of mobile terminals N	5 (small), 20 (large)
Number of SBSs K	3 (small), 4 (large)
Region radius R	200 m
Mobility intensity σ_m	2.0
LoS/NLoS transition probabilities (p_{LL}, p_{NN})	(0.95, 0.90)
System bandwidth B	1.0 MHz
Maximum transmit power $P_{\max}$	0.2 W
Local computing capacity F_{loc}	1.0 Gcycles/s
Macro-edge computing capacity F_0	8.0 Gcycles/s
SBS computing capacity F_{SBS}	4.0 Gcycles/s
CPU cycles per bit $C_n(t)$	800 cycles/bit
Effective capacitance coefficient κ	1×10^{-28}
Reward weights (w_d, w_e, w_q, w_o)	(1.0, 0.05, 0.2, 0.5)

Two representative MARL baselines are selected in this study for a controlled comparison. MASAC is used as the direct backbone baseline because the proposed method is developed by improving the MASAC learning framework. This comparison helps evaluate whether the proposed bounded action modeling, adaptive entropy tuning, and delayed actor update can improve the original MASAC backbone under the same state representation, action parameterization, reward definition, training budget, and evaluation protocol. MAPPO is adopted as another representative on-policy MARL baseline based on a different policy-optimization paradigm.

It should be noted that the current comparison is not intended to serve as an exhaustive benchmark of all continuous-action or hybrid-action MARL algorithms for MEC. Recent hybrid-action MARL methods usually rely on different action decompositions, discrete-continuous coupling mechanisms, state definitions, and resource-allocation assumptions. Directly adapting these methods to the present relaxed-action CTDE

environment would require additional redesign and implementation choices, which may introduce uncontrolled factors beyond the scope of this revision. Therefore, the experiments focus on a controlled comparison with the direct MASAC backbone and a representative on-policy MARL baseline. Broader comparisons with hybrid-action and continuous-action MARL methods under a unified MEC environment will be treated as an important direction for future work. Each method is trained for 50,000 steps, evaluated every 100 steps, and averaged over three seeds; Table 2 lists common settings.

Table 2: Common training settings.

Parameter	Value
Total training steps T	50,000
Evaluation interval ΔT_{eval}	100 steps
Replay buffer capacity $\lvert\mathcal{D}\rvert$	100,000
Mini-batch size B	1024
Discount factor γ	0.99
Soft update rate τ	0.01
Actor learning rate η_π	3×10^{-4}
Critic learning rate η_Q	3×10^{-4}
Policy delay d	2
Adaptive entropy tuning (α)	Enabled
Number of training seeds N_{seed}	3
Reward clipping interval $[r_{\min}, r_{\max}]$	$[-10, 0]$

Table 3 summarizes implementation details. SAC-based methods use decentralized actors and centralized twin critics under CTDE; ABDMASAC uses a Beta policy, whereas MASAC uses a tanh-squashed Gaussian policy. Other settings are kept identical unless specified.

Table 3: Implementation details of ABDMASAC and learning baselines.

Item	Setting
Actor network	Two fully connected hidden layers
Critic network	Centralized twin critics with two hidden layers
Hidden units	256 and 256
Activation function	ReLU
ABDMASAC policy	Beta policy
Beta parameter transform	Softplus transformation with lower-bound clipping
Beta concentration range	$[10^{-3}, 100]$
MASAC baseline policy	Tanh-squashed Gaussian policy
Gaussian log-standard-deviation range	$[-5, 2]$
Optimizer	Adam
Learning rate	3×10^{-4}
Discount factor γ	0.99
Soft update coefficient τ	0.01
Replay buffer capacity	100,000

(Continued)

Table 3 (continued)

Item	Setting
Warm-up steps	2000
Learning-start step	2000
Mini-batch size	1024
Learning frequency	Every environment step
Initial entropy coefficient α	0.2
Target entropy	$-\dim(\mathcal{A}_n)$
Entropy-coefficient range	$[10^{-4}, 10]$
Adaptive entropy tuning	Enabled for ABDMASAC
Policy-delay interval	2
Target-network update interval	2
Critic loss	Huber loss
Gradient clipping threshold	10.0
Target-Q clipping threshold	10^6
Reward clipping range	$[-10, 0]$
Evaluation interval	100 training steps
Evaluation episodes	3
Evaluation steps per episode	200
Evaluation seed set	1000–1009
Device	CUDA if available; otherwise CPU

Reward weights and target entropy are fixed across methods. The metrics are average return, end-to-end latency, energy consumption, and, for ablation, queue length.

4.2 Performance Comparison

4.2.1 Results in the Small-Scale Scenario

Fig. 3 shows that ABDMASAC obtains the best overall trade-off in the small-scale scenario. Its final return is about −1.85, better than MASAC (−2.75) and MAPPO (−2.40), and its latency is about 0.16 s, lower than MASAC (0.23 s) and MAPPO (0.20 s). MAPPO has the lowest energy, about 0.0112 vs. 0.0128 for ABDMASAC, but this comes with worse return and latency.

4.2.2 Results in the Large-Scale Scenario

Fig. 4 shows a clearer advantage at larger scale. ABDMASAC reaches about −1.02 return, 0.082 s latency, and 0.0050 energy, compared with MASAC (−1.24, 0.100 s, 0.0056) and MAPPO (−1.25, 0.101 s, 0.0074). Thus, ABDMASAC improves return while reducing both latency and energy consumption under stronger multi-agent competition.

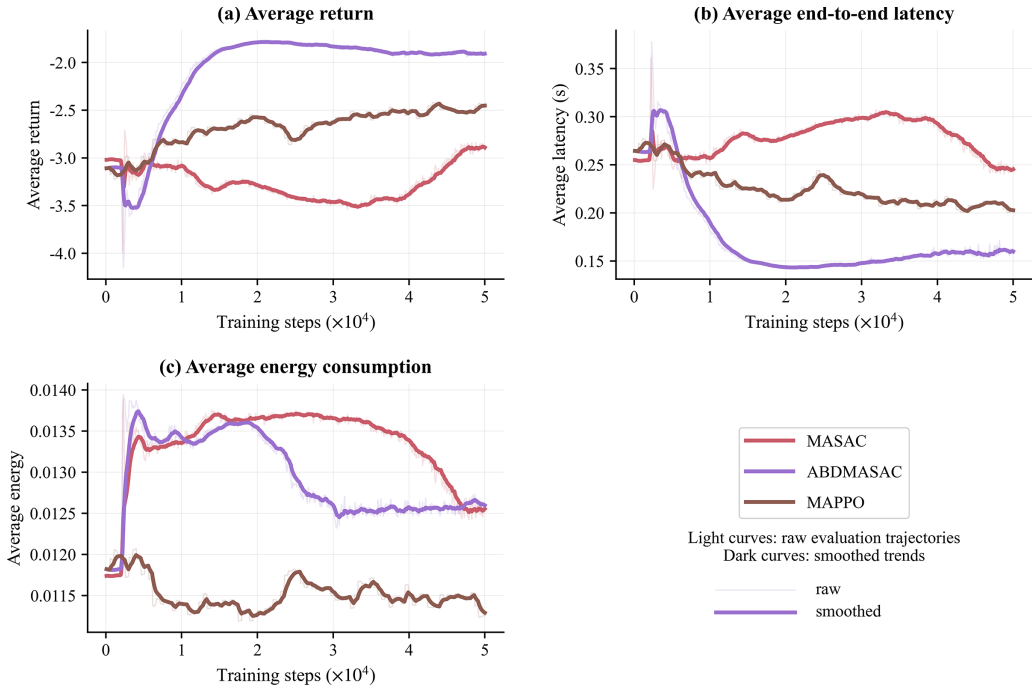

Figure 3: Performance comparison in the small-scale scenario: (**a**) average return, (**b**) average end-to-end latency, and (**c**) average energy consumption. The light curves denote the raw evaluation trajectories, while the dark curves denote the smoothed trends.

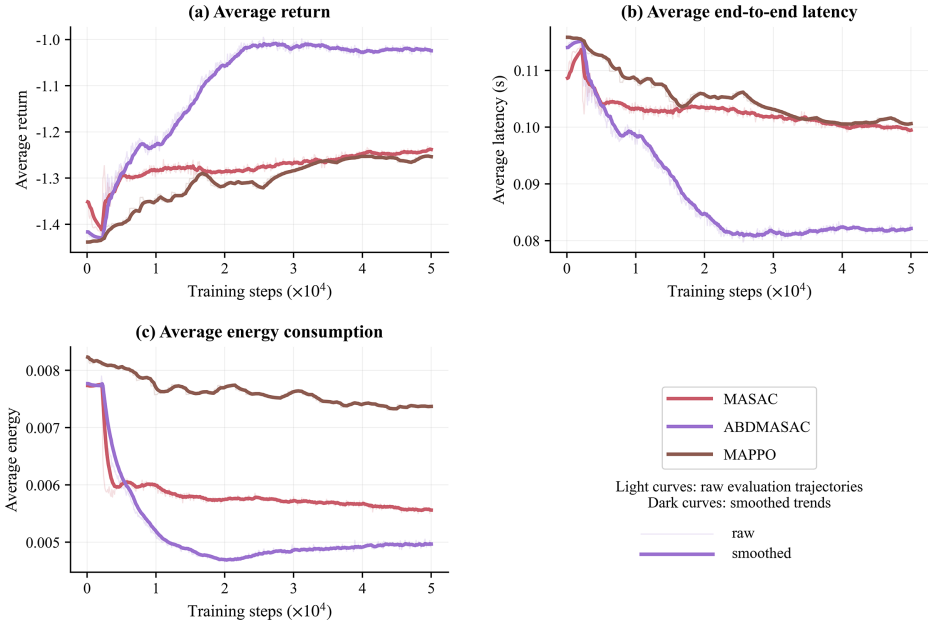

Figure 4: Performance comparison in the large-scale scenario: (**a**) average return, (**b**) average end-to-end latency, and (**c**) average energy consumption. The light curves denote the raw evaluation trajectories, while the dark curves denote the smoothed trends.

4.3 Ablation Study

4.3.1 Ablation Settings

A reduced environment is used for ablation while retaining the main load and resource-competition pattern. Four variants are tested: Baseline (standard MASAC), Beta-only, Beta-adaptive, and Full (ABD-MASAC). Table 4 lists their settings; means and standard deviations are reported over multiple seeds.

Table 4: Implementation settings of ablation variants.

Variant	Beta Policy	Adaptive Entropy Tuning	Policy-Delay Interval
Baseline	No, replaced by tanh-Gaussian policy	No, fixed $\alpha = 0.2$	1
Beta-only	Yes	No, fixed $\alpha = 0.2$	1
Beta-adaptive	Yes	Yes	1
Full (ABDMASAC)	Yes	Yes	2

4.3.2 Ablation Results and Discussion

Figs. 5 and 6 show that Baseline obtains about −3.28 return, while Beta-only decreases to about −3.73, with latency increasing to 0.321 s and queue length to 0.67×10^{-4}. The Beta policy mainly provides a bounded action representation, but using it alone may worsen learning because the exploration strength and update dynamics are not properly adjusted. Beta-adaptive gives the best return (−2.13), latency (0.183 s), and queue length (0.22×10^{-4}), with energy about 0.0090. Full further lowers energy to 0.0088, but its return, latency, and queue length are about −2.50, 0.231 s, and 0.49×10^{-4}, respectively. Thus, adaptive entropy tuning provides the clearest gain, while delayed updates adjust the final trade-off; the modules are complementary rather than independently monotonic.

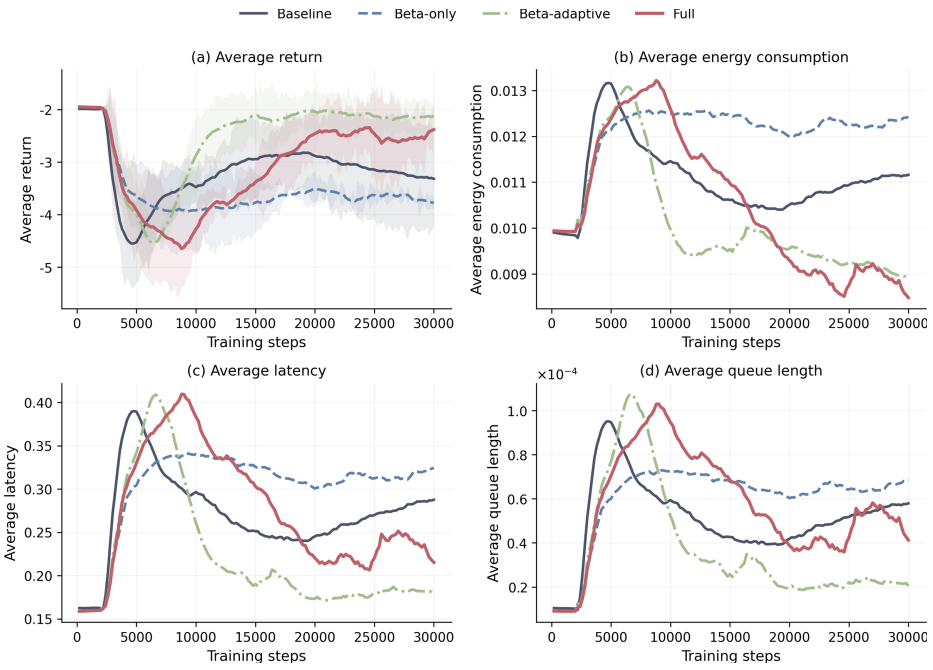

Figure 5: Training-stage metric trajectories of different ablation variants under the Ablation-XS setting: (**a**) reward, (**b**) energy, (**c**) latency, and (**d**) queue length.

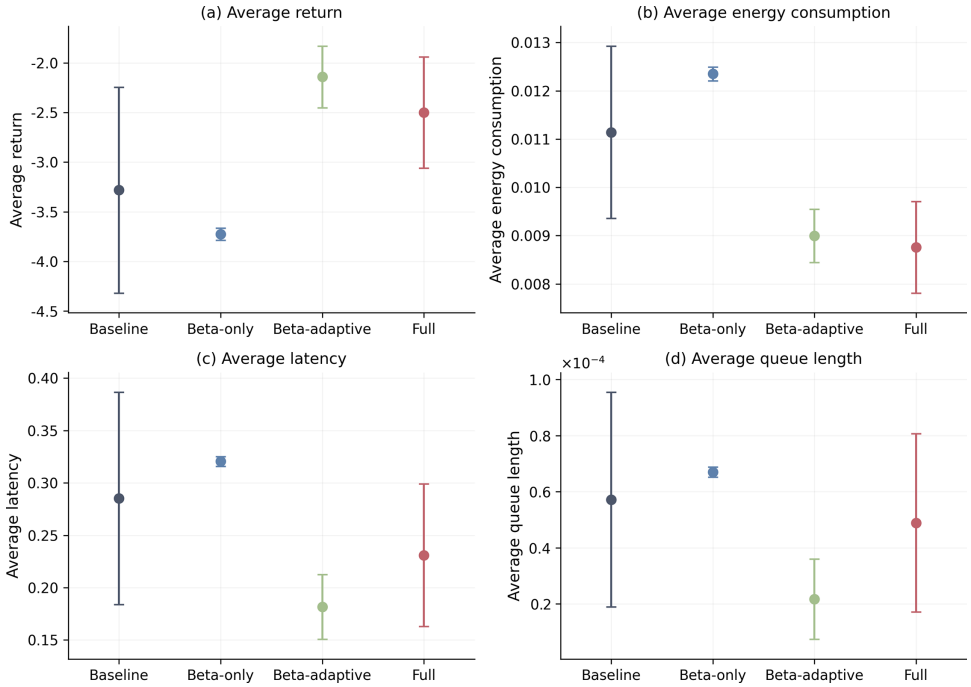

Figure 6: Statistical comparison over the last 20 evaluation points in the late training stage.

5 Conclusion

This paper investigates cooperative task offloading in dynamic MEC networks under CTDE and proposes ABDMASAC, which integrates Beta-policy-based bounded action modeling, adaptive entropy regulation, and delayed policy updates. Under the considered simulation settings, ABDMASAC achieves a better overall trade-off than the selected MASAC-backbone and on-policy MARL baselines. The ablation results show that the modules are complementary but not independently monotonic: the Beta policy alone may degrade performance when exploration and update dynamics are not properly adjusted, whereas the full method benefits from coordinated integration. The current comparison is not an exhaustive benchmark of all hybrid-action or continuous-action MARL methods. Future work will consider broader MARL comparisons, heterogeneous tasks, dynamic bandwidth allocation, communication overhead, explicit edge-server queueing, and security- and privacy-aware offloading.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Youth Talent Project of the Scientific Research Program of the Yunnan Key Research and Development Program under Grant 202503AP140020, the Major Science and Technology Special Project of Yunnan Province under Grant 202502AD080003, the Hubei Provincial Department of Education under Grant Q20241809, and the Doctoral Scientific Research Foundation of Hubei University of Automotive Technology under Grant BK202404.

Author Contributions: Zheng Yao and Changjun Deng: conceptualization. Zheng Yao, Jie Liu and Changjun Deng: data curation; investigation. Zheng Yao and Jie Liu: writing—original draft. Zheng Yao, Jie Liu, Changjun Deng, and Wang Lin: writing—review & editing. All authors reviewed and approved the final version of the manuscript.

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

Ethics Approval: Not applicable.

Conflicts of Interest: Changjun Deng and Wang Lin are affiliated with Yunnan Hanzhe Technology Co., Ltd. through the Yunnan International Joint Laboratory of Agricultural Remote Sensing and Digital Technology. This affiliation may be perceived as a potential conflict of interest. The authors declare that this affiliation did not influence the study design, data analysis, interpretation of results, manuscript preparation, or decision to publish. The remaining authors declare no conflicts of interest.

References

1. Chen X, Jiao L, Li W, Fu X. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw. 2016;24(5):2795–808. doi:10.1109/tnet.2015.2487344.
2. Mao Y, Zhang J, Song SH, Letaief KB. Power-delay tradeoff in multi-user mobile-edge computing systems. In: Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM); 2016 Dec 4–8; Washington, DC, USA. p. 1–6.
3. Wu Y, Qian LP, Ni K, Zhang C, Shen XS. Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading. IEEE J Sel Top Signal Process. 2019;13(3):392–407. doi:10.1109/jstsp.2019.2893057.
4. Yang W, Liu Z, Liu X, Ma Y. Deep reinforcement learning-based low-latency task offloading for mobile-edge computing networks. Appl Soft Comput. 2024;166(1):112164. doi:10.1016/j.asoc.2024.112164.
5. Li B, Liu W, Xie W, Li X. Energy-efficient task offloading and trajectory planning in UAV-enabled mobile edge computing networks. Comput Netw. 2023;234(1):109940. doi:10.1016/j.comnet.2023.109940.
6. Yao Z, Zhu Q, Zhang Y, Huang H, Luo M. Minimizing long-term energy consumption in RIS-assisted AAV-enabled MEC network. IEEE Internet Things J. 2025;12(12):20942–58. doi:10.1109/jiot.2025.3545252.
7. Gao H, Li W, Banez RA, Han Z, Poor HV. Mean field evolutionary dynamics in dense-user multi-access edge computing systems. IEEE Trans Wirel Commun. 2020;19(12):7825–35. doi:10.1109/twc.2020.3016695.
8. Tong Z, Wang J, Mei J, Li K. Multi-type task offloading for wireless internet of things by federated deep reinforcement learning. Future Gener Comput Syst. 2023;145(11):536–49. doi:10.1016/j.future.2023.04.004.
9. Han C, Chai Z, Li Y. Distributed task offloading in edge computing: a multi-objective adaptive deep reinforcement learning algorithm. Eng Appl Artif Intell. 2025;162:112653.
10. Misha T, Sun L, Chai ZY. Multi-objective multi-workflow task offloading based on evolutionary optimization. J King Saud Univ Comput Inf Sci. 2025;37(7):189. doi:10.1007/s44443-025-00073-8.
11. Yao Z, Chang P. High-dimensional multi-objective computation offloading for MEC in serial isomerism tasks via flexible optimization framework. Comput Mater Contin. 2026;86(1):1–18. doi:10.32604/cmc.2025.068248.
12. Mao Y, You C, Zhang J, Huang K, Letaief KB. A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tutor. 2017;19(4):2322–58. doi:10.1109/comst.2017.2745201.
13. Feng C, Han P, Zhang X, Yang B, Liu Y, Guo L. Computation offloading in mobile edge computing networks: a survey. J Netw Comput Appl. 2022;202(1):103366. doi:10.1016/j.jnca.2022.103366.
14. Kar B, Yahya W, Lin YR, Ali A. Offloading using traditional optimization and machine learning in federated cloud-edge-fog systems: a survey. IEEE Commun Surv Tutor. 2023;25(2):1199–226. doi:10.1109/comst.2023.3239579.
15. Dong S, Tang J, Abbas K, Hou R, Kamruzzaman J, Rutkowski L, et al. Task offloading strategies for mobile edge computing: a survey. Comput Netw. 2024;254(6):110791. doi:10.1016/j.comnet.2024.110791.
16. Wang D, Bin Abu Bakar K, Isyaku B, Eisa TAE, Abdelmaboud A. A comprehensive review on internet of things task offloading in multi-access edge computing. Heliyon. 2024;10(9):e29916. doi:10.1016/j.heliyon.2024.e29916.
17. Luo Z, Dai X. Reinforcement learning-based computation offloading in edge computing: principles, methods, challenges. Alex Eng J. 2024;108(6):89–107. doi:10.1016/j.aej.2024.07.049.
18. Hortelano D, de Miguel I, Duran Barroso RJ, Aguado JC, Merayo N, Ruiz L, et al. A comprehensive survey on reinforcement-learning-based computation offloading techniques in edge computing systems. J Netw Comput Appl. 2023;216(3):103669. doi:10.1016/j.jnca.2023.103669.

19. Zabihi Z, Eftekhari Moghadam AM, Rezvani MH. Reinforcement learning methods for computation offloading: a systematic review. ACM Comput Surv. 2024;56(1):1–41. doi:10.1145/3603703.
20. Peng P, Lin W, Wu W, Zhang H, Peng S, Wu Q, et al. A survey on computation offloading in edge systems: from the perspective of deep reinforcement learning approaches. Comput Sci Rev. 2024;53:100656.
21. Nabi A, Baidya T, Moh S. Comprehensive survey on reinforcement-learning-based task offloading techniques in aerial edge computing. Internet Things. 2024;28(3):101342. doi:10.1016/j.iot.2024.101342.
22. Tang M, Wong VWS. Deep reinforcement learning for task offloading in mobile edge computing systems. IEEE Trans Mob Comput. 2022;21(6):1985–97. doi:10.1109/tmc.2020.3036871.
23. Mirza MA, Yu J, Raza S, Krichen M, Ahmed M, Khan WU, et al. DRL-assisted delay optimized task offloading in automotive-industry 5.0 based VECNs. J King Saud Univ Comput Inf Sci. 2023;35(6):101512. doi:10.1016/j.jksuci.2023.02.013.
24. Liu J, Mi Y, Zhang X, Li X. Task graph offloading via deep reinforcement learning in mobile edge computing. Future Gener Comput Syst. 2024;158(1):545–55. doi:10.1016/j.future.2024.04.034.
25. Liu X, Chai ZY, Li YL, Cheng YY, Zeng Y. Multi-objective deep reinforcement learning for computation offloading in UAV-assisted multi-access edge computing. Inf Sci. 2023;642(1):119154. doi:10.1016/j.ins.2023.119154.
26. Xie M, Ye J, Zhang G, Ni X. Deep reinforcement learning-based computation offloading and distributed edge service caching for mobile edge computing. Comput Netw. 2024;250(2):110564. doi:10.2139/ssrn.4725151.
27. Mustafa E, Shuja J, Rehman F, Namoun A, Ali M, Alourani A. Deep reinforcement learning with dueling DQN for partial computation offloading and resource allocation in mobile edge computing. IEEE Access. 2025;13:94319–35. doi:10.1109/access.2025.3573929.
28. Tan S, Chen B, Liu D, Zhang J, Hanzo L. Communication-assisted multi-agent reinforcement learning improves task-offloading in UAV-aided edge-computing networks. IEEE Wirel Commun Lett. 2023;12(12):2233–7. doi:10.1109/lwc.2023.3316794.
29. Ju T, Li L, Liu S, Zhang Y. A multi-UAV assisted task offloading and path optimization for mobile edge computing via multi-agent deep reinforcement learning. J Netw Comput Appl. 2024;229(6):103919. doi:10.1016/j.jnca.2024.103919.
30. Shi H, Tian Y, Li H, Huang J, Shi L, Zhou Y. Task offloading and trajectory scheduling for UAV-enabled MEC networks: an MADRL algorithm with prioritized experience replay. Ad Hoc Netw. 2024;154(1):103371. doi:10.1109/lwc.2021.3122957.
31. Wang J, Zhang M, Yin Q, Yin L, Peng Y. Multi-agent reinforcement learning for task offloading with hybrid decision space in multi-access edge computing. Ad Hoc Netw. 2025;166(11):103671. doi:10.1016/j.adhoc.2024.103671.
32. Yao Z, Chang P, Khalil FK, Duan C. Joint node selection and task offloading via evolutionary game and MATD3 in UAV-assisted MEC networks. J King Saud Univ Comput Inf Sci. 2025;37(8):220. doi:10.1007/s44443-025-00248-3.
33. Chen J, Zhong J, Wu Z, Tian D, Chen Y. Priority-aware task offloading for LEO satellite edge computing network: a multi-agent deep reinforcement learning-based approach. J King Saud Univ Comput Inf Sci. 2025;37(7):168. doi:10.1007/s44443-025-00160-w.
34. Zuo P, Miao C, Fu C, Wang X, Liu X, Liu B. SMAPPO: a security-aware multi-agent reinforcement learning framework for secure computation offloading in SAGIN. J King Saud Univ Comput Inf Sci. 2025;37:336.
35. Li J, Li J, Shi Y, Lian H, Wu H. A multi-agent deep reinforcement learning algorithm for task offloading in future 6G V2X network. IEICE Trans Inf Syst. 2025;E108-D(7):697–708. doi:10.1587/transinf.2024iip0005.
36. Bo J, Zhao X. Vehicle edge computing task offloading strategy based on multi-agent deep reinforcement learning. J Grid Comput. 2025;23(2):13. doi:10.1007/s10723-025-09800-x.
37. Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017); 2017 Dec 4–9; Long Beach, CA, USA.
38. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the 35th International Conference on Machine Learning (ICML); 2018 Jul 10–15; Stockholm, Sweden.

39. Fujimoto S, van Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proceedings of the 35th International Conference on Machine Learning (ICML); 2018 Jul 10–15; Stockholm, Sweden. p. 1587–96.
40. Chou PW, Maturana D, Scherer S. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In: Proceedings of the 34th International Conference on Machine Learning (ICML); 2017 Aug 6–11; Sydney, Australia. p. 834–43.