Computing (2025) 107:126
https://doi.org/10.1007/s00607-025-01472-5

REGULAR PAPER

Task offloading and resource allocation for multi-UAV asset edge computing with multi-agent deep reinforcement learning

Samah A. Zakaryia[1] · Mohamed Meaad[2] · Tamer Nabil[3] · Mohamed K. Hussein[2]

Received: 4 January 2025 / Accepted: 24 March 2025

Abstract
Mobile edge computing (MEC) has emerged as a key solution for addressing the demands of computation-intensive network services by providing computational resources at the network edge, thereby minimizing service delays. Leveraging their flexible deployment, wide coverage, and reliable wireless communication, unmanned aerial vehicles (UAVs) have been integrated into MEC systems to enhance performance. This paper investigates the task offloading problem in a Multi-UAV-assisted MEC environment and proposes a collaborative optimization framework that integrates the Distance to Task Location and Capability Match (DTLCM) mechanism with a Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm. Unlike traditional task priority-based offloading schemes, the proposed approach ensures optimal UAV selection based on both computational capability and spatial proximity. The system gain is defined in terms of energy efficiency and task delay with the optimization formulated as a mixed-integer programming problem. To efficiently solve this complex problem, a Multi-Agent Deep Reinforcement Learning framework is employed, combining MADDPG with DTLCM to jointly optimize UAV trajectories, task offloading decisions, computational resource allocation, and communication resource management. Comprehensive simulations demonstrate that the proposed MADDPG-DTLCM framework significantly outperforms four state-of-the-art methods (MADDPG-DTLCM,MADQN, MADDPG without DTLCM, and Greedy offloading), achieving 18% higher task completion rates and 12% lower latency under varying network conditions, particularly in high-user-density scenarios with UAV collaboration.

Keywords Mobile edge computing · Task offloading · Multi-UAV network · Multi-agent deep reinforcement learning · Distance to task location and capability match

Mathematics Subject Classification 90B35 · 90C27 · 90C59 · 68M14 · 68T20

Extended author information available on the last page of the article

Published online: 03 May 2025

1 Introduction

The rapid evolution of modern network services has resulted in an increasing demand for substantial computational resources and ultra-low latency. Multi-Access or Mobile Edge Computing (MEC) has emerged as a critical paradigm to address these requirements by bringing computational capabilities closer to the network edge, thereby reducing latency. However, current MEC implementations face challenges in scenarios with high user density or sparse network infrastructure [1]. To overcome these limitations, Unmanned Aerial Vehicles (UAVs) have been proposed as a complementary solution for enhancing MEC systems due to their flexibility in deployment and wide-area coverage. UAV-assisted MEC systems present a promising approach for managing distributed, computation-intensive tasks [2].

Unlike traditional terrestrial MEC networks, where edge nodes are stationary and provide limited coverage confined to nearby users, UAV-assisted MEC offers dynamic deployment capabilities, allowing for extended service range and improved adaptability [3]. While task offloading with a single UAV has been extensively studied, its limited resources make it suitable only for small-scale tasks, and its potential to enhance performance is inherently constrained. Moreover, a single UAV is often insufficient to meet the growing computational demands of modern applications. This limitation highlights the need for Multi-UAV MEC systems, where multiple UAVs collaboratively provide extensive computational resources and service coverage [4].

Despite their advantages, Multi-UAV MEC systems introduce several challenges. These include designing UAV trajectories to prevent collisions, managing communication resources efficiently to enhance data transmission, and developing strategies for collaborative task offloading to distribute the computational workload effectively. Many existing UAV-assisted task offloading strategies emphasize partial offloading, which involves partitioning tasks into smaller sub-tasks. This method provides flexibility in resource allocation, reduces processing delays, and is particularly effective for divisible tasks [5, 6].

However, partial offloading is not suitable for indivisible tasks. In such cases, binary offloading, which involves fully offloading tasks or executing them locally, can serve as a viable alternative. While binary offloading adds another dimension to task management, it increases complexity by introducing a joint optimization problem involving both continuous and discrete variables. Furthermore, UAV task requests and resource demands are highly dynamic, making the optimization process even more challenging. Achieving long-term average optimization in this non-convex, time-varying environment with incomplete future information requires advanced methodologies [7, 8].

Recent advancements in artificial intelligence (AI), particularly Deep Reinforcement Learning (DRL), have demonstrated significant potential in addressing long-term optimization problems [9]. DRL, by leveraging historical data and dynamically exploring the environment, enables intelligent decision-making under uncertainty. This approach facilitates the acquisition of optimal long-term average rewards, making it a powerful tool for addressing the challenges inherent in UAV-assisted multi-MEC systems [10].

To address the challenges outlined earlier, this paper investigates task offloading in collaborative UAV-assisted Mobile Edge Computing (MEC) systems by integrating the Distance to Task Location and Capability Match (DTLCM) mechanism. Unlike existing UAV-assisted MEC studies, which primarily focus on isolated optimizations for trajectory planning or task scheduling, our framework jointly optimizes UAV movement, resource allocation, and task distribution to ensure long-term system performance improvements.

The proposed framework leverages Multi-Agent Deep Deterministic Policy Gradient (MADDPG) for task offloading optimization due to its ability to handle continuous action spaces while maintaining Centralized Training with Decentralized Execution (CTDE), ensuring coordinated UAV task allocation. Unlike MAPPO, which assumes full state observability, MADDPG employs a centralized critic that enables efficient agent coordination without requiring a complete global state. Furthermore, compared to MATD3, MADDPG prevents overestimation bias and ensures smoother policy updates, leading to more stable training and better convergence in dynamic MEC environments.

By formulating the problem as a Markov Decision Process (MDP), our approach enables intelligent UAV decision-making in dynamic and uncertain environments. A novel Deep Reinforcement Learning (DRL) framework is introduced to tackle the complexities of multi-UAV collaboration, ensuring optimal long-term system gain. The main contributions of the paper are as follows:

1 A Novel MADDPG-Based Framework for UAV-Assisted MEC: We introduce a Multi-Agent Deep Deterministic Policy Gradient (MADDPG) framework, specifically designed for task offloading and resource allocation in multi-UAV-assisted MEC systems. Unlike traditional heuristic-based or decentralized MARL methods, our framework incorporates a centralized critic for enhanced coordination, resulting in efficient UAV positioning, task allocation, and computational resource utilization.
2 A Novel Distance to Task Location and Capability Match (DTLCM) Mechanism: The DTLCM mechanism ensures that tasks are assigned to UAVs that are both optimally positioned and computationally capable of handling the workload. This dual-metric approach significantly enhances resource utilization, reduces offloading delays, and balances computational load across UAVs, unlike conventional distance-only-based task allocation strategies.
3 Joint Optimization of UAV Trajectories, Task Offloading, and Resource Management: We formulate the task offloading problem as a Multi-Agent Markov Decision Process (MDP), enabling UAVs to adaptively optimize trajectories, task scheduling, and transmission power. The proposed model accounts for the stochastic nature of UAV positions, dynamic task arrivals, computational demands, ensuring robust, and efficient system performance.
4 A Comprehensive Long-Term Optimization Approach: Our framework introduces a holistic optimization model that simultaneously considers UAV mobility,

offloading strategies, transmission power allocation, and computational resource distribution. Unlike previous studies that treat these components separately, our method jointly optimizes all factors, achieving lower latency, higher energy efficiency, and seamless UAV collaboration, while maintaining collision avoidance and network coverage constraints.

5 Extensive Performance Evaluation and Comparison with State-of-the-Art Approaches: We conduct extensive simulations to evaluate the proposed MADDPG-DTLCM framework against benchmark algorithms, including MADQN and greedy offloading strategies. The results demonstrate significant improvements in system gain, task completion rates, and overall efficiency, particularly under high user density and computational demand conditions.

2 Related work

To provide a structured overview of existing research, we categorize prior work into three key areas:

2.1 UAV-assisted task offloading approaches

Task offloading in UAV-assisted Mobile Edge Computing (MEC) environments has been extensively studied to optimize task distribution, execution delay, and energy efficiency.

The authors in [11] address the challenge of efficiently managing computation and resource allocation in IoT networks assisted by multiple UAVs. Their Multi-Agent Deep Reinforcement Learning (MADRL) approach aims to minimize network computation costs while optimizing Quality of Service (QoS) for IoT devices. By modeling the system as a stochastic game, UAVs act as agents making decentralized, cooperative decisions. Results show that this approach effectively enhances resource utilization and reduces computation delays in dynamic IoT settings. Similarly, the authors in [12] focuses on efficient task distribution in edge networks supported by UAVs. By employing a Multi-Agent Deep Reinforcement Learning (DRL) framework, the study optimizes offloading strategies to reduce energy consumption and minimize latency. UAVs, acting as intelligent agents, collaboratively decide on task allocation, communication resource management, and flight paths to achieve system-wide efficiency and enhanced service quality for mobile edge devices. The authors in [13] explore Non-Orthogonal Multiple Access (NOMA) techniques to improve IoT connectivity and task scheduling efficiency. While this approach enhances network communication, it does not explicitly optimize UAV task allocation, making it applicable to task offloading strategies in UAV-MEC systems. Furthermore, [14] investigates task offloading challenges in UAV-assisted edge computing for IoT networks. This study highlights the importance of optimizing UAV trajectory planning to enhance task offloading efficiency and balance computational loads among UAVs. Unlike traditional approaches relying on centralized controllers, which incur high communication costs and limited agent coordination, this paper introduces a communication-assisted, decentralized Multi-Agent Reinforcement Learning (MARL) framework, allowing UAVs to exchange critical information for better distributed decision-making. Task offloading in UAV-assisted Mobile Edge Computing (MEC) has gained significant attention due to the growing demand for low-latency and energy-efficient processing. The authors in [15] Proposed an energy-efficient multi-stage alternating optimization scheme for UAV-mounted MEC networks. Their work optimizes task allocation and computational resource scheduling but does not fully integrate Multi-Agent Reinforcement Learning (MARL) techniques. Our study extends this research by leveraging a DTLCM-based MARL framework to enhance task scheduling and UAV coordination dynamically.

2.2 Multi-agent reinforcement learning in MEC systems

With the increasing complexity of UAV-assisted MEC networks, Multi-Agent Reinforcement Learning (MARL) has been widely adopted to enhance task offloading, trajectory planning, and resource allocation.

For example, [16] proposes MARS, a Multi-Agent DRL-based framework for enhancing resource scheduling in a UAV and Intelligent Reflecting Surface (IRS)-assisted MEC environment. This system enables UAVs with MEC servers to collaborate with IRSs, which are strategically placed to mitigate signal blockages and improve communication quality in urban settings. The author in [17] presents a DRL framework for task offloading in the Internet of Vehicles (IoV) environment. By integrating the Hungarian algorithm with a Multi-Agent DRL approach, the study allows moving and stationary vehicles to act as collaborative fog nodes, optimizing task allocation based on real-time network conditions. Additionally, [18] proposes a MARL-based task offloading system for UAV-assisted MEC, where UAVs cooperate to enhance computational efficiency for IoT devices. The study demonstrates how UAVs cooperatively make decisions regarding task distribution and flight trajectories to balance computational loads, minimize latency, and reduce energy consumption in dynamic network environments. Further, [19] introduces TD3-TT, a Twin Delayed Deep Deterministic Policy Gradient (TD3) model for optimizing UAV trajectory and task offloading in MEC systems. The approach integrates Directed Acyclic Graph (DAG) scheduling to significantly reduce task execution delays. Simulation results confirm that TD3-TT outperforms conventional UAV-based offloading methods in terms of efficiency, convergence speed, and adaptability in real-time MEC scenarios.

Existing research has applied multi-agent reinforcement learning (MARL) to UAV coordination in MEC environments [20]. Incorporated a decentralized task management system for UAVs but did not consider dynamic resource allocation and UAV mobility optimization, which our approach addresses using the DTLCM mechanism and MADDPG framework.

2.3 Resource optimization techniques for UAVs

Optimizing UAV trajectory planning and computational resource allocation remains a crucial aspect of UAV-assisted MEC systems.

The authors in [21] explore Multi-Agent DRL for task offloading in Vehicular Edge Computing (VEC) networks, where task-intensive computations are offloaded from vehicles to edge servers. Their framework aims to reduce latency and maximize throughput by dynamically managing offloading resources based on real-time network conditions. Similarly, [22] proposes a Deep Deterministic Policy Gradient (DDPG)-based method to optimize UAV trajectory and computation offloading in MEC systems. The study addresses key challenges related to task scheduling and UAV mobility, utilizing K-nearest neighbor (KNN) and prioritized experience replay to improve learning efficiency and convergence stability. Moreover, the author in [2] introduces a robust framework for optimizing task offloading and trajectory planning in UAV-assisted MEC environments, considering uncertainties in communication channels and task complexity. Unlike conventional approaches that assume idealized network conditions, this study provides a realistic evaluation of UAV performance in highly dynamic environments. Additionally, the author in [23] investigates how deep learning methods can enhance task offloading in UAV-supported MEC systems. This research highlights the flexibility of UAVs as mobile computing nodes, capable of managing high-demand computational tasks from ground-based users while reducing latency and improving computational efficiency. The author in [24] explores Distributed Machine Learning (DML) techniques for resource scheduling in edge computing environments. This approach enhances resource utilization but does not specifically consider UAV mobility or offloading decisions. Similarly, [25] proposes Federated Learning (FL)-based communication models for optimizing computational resources in edge networks. While this work is highly relevant to MEC, it primarily focuses on static ground-based networks rather than UAV-assisted MEC. Finally [26] explores MADRL-based optimization for UAV-aided MEC environments, focusing on task offloading and trajectory decisions. By enabling UAVs to learn collaborative policies, this study enhances resource allocation and energy efficiency, allowing UAVs to autonomously adapt to dynamic network conditions. Optimizing UAV resources in MEC systems requires balancing computational capacity, energy efficiency, and network constraints. While [27] focused on IoT machine learning job placement in fog computing, our study integrates a DTLCM-based reinforcement learning framework to optimize real-time UAV task scheduling, trajectory planning, and computational resource management under highly dynamic network conditions. While previous works [3–7] have explored UAV-assisted MEC using DRL techniques, they primarily focus on either trajectory optimization or task offloading separately. In contrast, our approach jointly optimizes UAV movement, task scheduling, and computation resource allocation in a fully integrated DTLCM-enhanced MADDPG framework. Moreover, we explicitly address the challenge of agent coordination, which is overlooked in decentralized MARL models like MAPPO and MATD3. A comparative analysis of existing task offloading strategies in UAV-assisted MEC environments highlights the limitations of approaches that focus exclusively on either task scheduling or trajectory optimization. In contrast, our method integrates MADDPG with DTLCM to achieve an optimal balance between task proximity and computational capability, as summarized in Table 1.

Table 1 Presents a comparative analysis of existing task offloading strategies in UAV-assisted MEC environments

Study	Technique Used	Dataset	Performance Metrics	Advantages	Disadvantages
[28]	Proximal Policy Optimization (PPO) for UAV Path Planning and Task Offloading	Simulated UAV and IoT dataset	Age of Information (AoI), Energy consumption, Path planning efficiency	Balances information freshness and energy efficiency, enhances UAV path planning	PPO-based models can be unstable in complex multi-UAV environments
[1]	Multi-Objective Reinforcement Learning for UAV Trajectory Optimization	Simulated UAV-MEC dataset	Energy efficiency, Task execution time, UAV lifetime	Incorporates wireless charging, optimizes multi-objective trajectory planning	Does not address real-time task offloading dynamics
[29]	Partial Task Offloading and Resource Allocation in Multi-UAV Networks	Simulated UAV networks (5G Advanced)	Computational delay, Energy efficiency, UAV trajectory optimization	Optimizes offloading under 5G connectivity, improves UAV cooperation	Does not fully integrate reinforcement learning for adaptive decision-making
[30]	Game Theory for Air-Ground Integrated Networks	Simulated UAV-MEC dataset	Energy consumption, Task success rate, Computational efficiency	Minimizes energy consumption, enhances distributed resource allocation	Game-theoretic approaches may struggle with real-time UAV dynamics
The proposed Work	Multi-Agent Deep Deterministic Policy Gradient (MADDPG) with Distance to Task Location and Capability Match (DTLCM) for UAV Task Offloading	Simulated UAV-MEC environment	Task latency, Energy efficiency, System gain, Task completion rate	Joint optimization of UAV trajectories, task offloading, and resource allocation; DTLCM improves task allocation	High computational complexity for training MARL models

Unlike previous works that focus solely on task scheduling or trajectory optimization, our approach integrates MADDPG with DTLCM, ensuring an optimal balance between task proximity and computational capability

3 System model and problem formulation

Figure 1 illustrates a Multi-UAV-assisted Mobile Edge Computing (MEC) system comprising V User Equipment (UDs) and N Unmanned Aerial Vehicles (UAVs). Each UE m periodically generates computationally intensive tasks, represented as $M_v = (D_v, C_v, \lambda_v)$, where D_v denotes the size of the task data, C_v signifies the required CPU cycles, and λ_v represents the task arrival rate. Due to their constrained computational resources, the UEs are unable to execute these tasks locally. Consequently, UAVs are deployed to provide MEC services to the ground-based UDs. The deployment of UAVs is meticulously planned to ensure non-overlapping trajectories, thereby conserving energy and preventing collisions.

3.1 UAVs movement

The 3D position of UAV n at time t is defined as $p_n(t) = [x_n(t), y_n(t), z_n(t)]^T$, where $x_n(t)$, $y_n(t)$, and $z_n(t)$ represent its coordinates along the X, Y, and Z axes, respectively. In the 2D plane, the UAV's position is expressed as $v_n(t) = [x_n(t), y_n(t)]^T$. It is assumed that UAV n moves a horizontal distance of $l_n(t)$ at an angle $\theta_n(t)$ within the range $[0, 2\pi)$. This leads to the following mathematical relationship:

$$x_n(t+1) = x_n(t) + l_n(t)cos\left(\theta_n(t)\right), \tag{1}$$

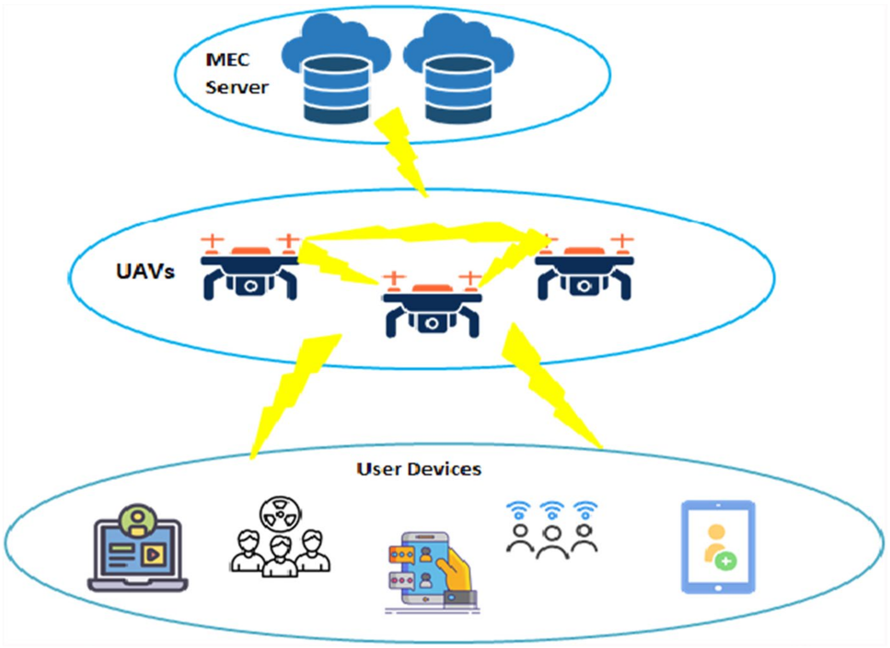

Fig. 1 Multi-UAV asset MEC

$$y_n(t+1) = y_n(t) + l_n(t)sin\left(\theta_n(t)\right), \tag{2}$$

Let us consider that UAV n operates with a maximum elevation angle denoted by α_n. At any given time t, this defines the maximum horizontal radius that UAV n can achieve, denoted as $R_n^{max}(t)$, can be calculated as:

$$R_{max}^n(t) = z_n(t)tan\left(\alpha_n\right), \tag{3}$$

The UAV movement follows a kinematic model, where the next position $z_n(t+1)$ is determined by:

$$z_n(t+1) = z_n(t) + v_n(t).\Delta t + \eta_n. \tag{4}$$

where $v_n(t)$ is the UAV velocity vector, Δt is the discrete time step, and η_n accounts for environmental uncertainty (e.g., wind disturbances).

This formulation ensures that UAVs update their positions based on real-time velocity control, preventing collisions while optimizing trajectory planning.

UAVs have restricted flight distances due to limitations on their horizontal and vertical speeds. This range can be described as:

$$Z_{min} \le Z_n(t) \le Z_{max}. \tag{5}$$

$$l_n(t) = \left|\left|v_n(t+1) - v_n(t)\right|\right| \le L_{\max}^h. \tag{6}$$

where $v_n(t)$ and $v_n(t+1)$ represent the velocity of UAV n at consecutive time steps, where each task has a specific computational requirement and offloading priority.

$$\Delta z_n(t) = \left|z_n(t+1) - z_n(t)\right| < L_{max}^v. \tag{7}$$

Here, Z_{min} and Z_{max} represent the lowest and highest permissible altitudes, respectively. $\Delta z_n(t)$ Refers to the vertical movement distance, while L_{max}^h and L_{max}^v indicate the maximum allowable distances for horizontal and vertical movement of the UAV. In order to prevent the overlap of coverage regions between any two UAVs, the following constraint on spatial separation must be enforced:

$$\left|\left|v_n(t) - v_j(t)\right|\right| \ge \left[R_{max}^n(t) + R_{max}^j(t)\right], \forall n, j, n \ne j. \tag{8}$$

Similarly, to prevent collisions between any two UAVs, the distance between them must be greater than or equal to a minimum threshold D_{min}. Therefore, the following collision avoidance constraint must be satisfied:

$$\left|\left|p_n(t) - p_j(t)\right|\right| \ge D_{min}, \quad \forall n, j, n \ne j. \tag{9}$$

3.2 Communication model

We define $p_v(t) = [x_v(t), y_v(t), 0]^T$ as the position of Ground user devices UD v, where $x_v(t)$ and $y_v(t)$ represent its X and Y coordinates, respectively. The distance between UAV n and UD m is expressed as:

$$d_{vn}(t) = ||p_n(t) - p_v(t)|| \tag{10}$$

UDs are assumed to communicate with their associated UAVs using Orthogonal Frequency-Division Multiple Access (OFDMA), effectively minimizing interference among UDs within the coverage area of each UAV. Given the significant altitude of UAVs, the channel is predominantly influenced by the Line-Of-Sight (LOS) component, with effects from shadowing and small-scale fading being negligible. Furthermore, any Doppler shift caused by the high mobility of UAVs is assumed to be perfectly compensated at the UDs, as detailed in [2].

In UAV-assisted MEC systems, the channel gain is typically modeled based on a Line-Of-Sight (LOS) transmission model due to the high altitude and clear signal propagation of UAVs. The path loss model is given as:

$$k_{vn}(t) = \frac{g_0}{[d_{vn}(t)]^2}. \tag{11}$$

where g_0 the reference is channel gain and $d_{vn}(t)$ represents the distance between UAV n and user v. This model effectively captures the signal attenuation behavior in airborne communication scenarios.

In the task offloading process, the uplink bandwidth B_u is evenly distributed among all user devices (UDs). Consequently, the ground-to-air (G2A) data rate between UD v and UAV n is given by:

$$J_{vn}(t) = \frac{B_u}{M_n(t)} \log_2 \left[1 + \frac{K_{vn}(t)pt_v}{\sigma_u^2}\right], \tag{12}$$

where $J_{vn}(t)$ is achievable transmission rate between UAV n and task v at time t, B_u is available bandwidth for UAV communication, $M_n(t)$ is number of users sharing UAV n's bandwidth at time t, $K_{vn}(t)$ is channel gain between UAV n and task v at time t, pt_v is transmission power allocated to task v, and σ_u^2 is additive white Gaussian noise (AWGN) power.

As all tasks are offloaded to UAVs through the ground-to-air (G2A) channel, the G2A transmission delay between a user device (UD) v and a UAV n is determined by dividing the task data size D_v by the corresponding transmission data rate $J_{vn}(t)$, as shown below:

$$T_{vn}^{G2A}(t) = \frac{D_v}{J_{vn}(t)}, \tag{13}$$

Similarly, the ground-to-air (G2A) transmission energy consumption between user device (UD) v and UAV n can be expressed as the product of the transmission power and the G2A transmission delay.

$$E_{vn}^{G2A}(t) = pr_n^r T_{vn}^{G2A}(t) = \frac{D_v pr_n^r}{J_{vn}(t)}. \tag{14}$$

where pr_n^r is referee the receiving power of UAV n.

3.3 Task classification model

Tasks assigned to UAVs are classified using a combination of two factors: the distance to the task location and the match between the UAV's capabilities and the task's requirements.

3.3.1 Distance to the task location (DTL)

This metric focuses on the spatial proximity between the UAV and the task location. The closer the UAV is to the task, the lower the communication latency and better the energy efficiency because the transmission distance is minimized.

The distance between the UAV and the task location is typically calculated using the Euclidean distance formula**:**

$$D_n = \sqrt{(x_n - x_t)^2 + (y_n - y_t)^2 + (z_n - z_t)^2}. \tag{15}$$

where x_n, y_n, z_n are the coordinates of UAV n, x_t, y_t, z_t are the coordinates of the task location.

Then calculate normalized distance to a range between 0 and 1 for easier comparison:

$$NormalizedDistance_n = \frac{D_n}{D_{max}}. \tag{16}$$

where D_n is the actual distance between the UAV and the task and D_{max} is the maximum allowable operational distance in the environment.

3.3.2 Capability match (CM)

By taking into account the capabilities of each UAV, tasks can be assigned to those with the necessary resources and expertise to perform the task effectively, leading to improved task execution quality and efficiency.

$$CM_n = \frac{ComputeResourcesofUAV_n}{RequiredComputeResourcesofTask_i}. \tag{17}$$

Equation (16) serves as a guideline for optimal task allocation rather than imposing a strict constraint. While the CM metric prioritizes UAVs with closely matched computational resources, UAVs with are still considered for task allocation.

The system ensures that UAVs with higher computational resources remain available for more demanding tasks while balancing overall resource utilization. The capabilities match score CM_n, which measures the adequacy of UAVn's available computational resources relative to the task's required compute power. Unlike the previous formulation, which assumed CM_n is always between 0 and 1, this revised definition allows for values greater than 1, indicating that a UAV has surplus computational resources.

If $CM_n < 1 \rightarrow$ The UAV has insufficient resources for the task.

If $CM_n = 1 \rightarrow$ The UAV's resources exactly match the task's requirements.

If $CM_n > 1 \rightarrow$ The UAV has excess computing resources, allowing for potential multi-tasking or more efficient load balancing.

Integration of CM into the Task Offloading Strategy the CM metric is incorporated into the DTLCM-based task classification, ensuring that UAVs with sufficient resources are prioritized for execution. The updated UAV-task matching score is defined as:

$$\arg \begin{matrix} \max \\ n \end{matrix} \left(\alpha_1 \frac{1}{d_{vn}(t)} + \alpha_2 CM_n \right).$$

where $d_{vn}(t)$ is refer represents the distance between UAV n and the task location, CM_n is represents the computational capability match, and α_1 and α_2 are weighting coefficients balancing the impact of distance and computational efficiency.

This adjustment better represents real-world UAV-MEC scenarios, where some UAVs have significant excess computational capacity and can handle multiple tasks efficiently.

3.3.3 Combining the two metrics

The two metrics Distance and Capability are combined to form a Suitability Score (SS)**,** which helps decide which UAV should handle a specific task. The Suitability Score is a weighted combination of the Normalized Distance and Capability Match (CM)**:**

$$SS_n = n_d.\left(1 - NormalizedDistance_n\right) + n_{cm}.CM_n. \tag{18}$$

where n_d the weight is assigned to the distance metric, reflecting how important proximity to the task is cm is the weight assigned to the capability match, reflecting how important the computational resources of the UAV are for the task, and $(1 - NormalizedDistance_n)$ ensures that closer UAVs get higher scores.

While the weighted sum of DTL and CM provides an effective heuristic for UAV selection, it does not guarantee optimality under all scenarios. Jointly optimizing UAV selection with task offloading and resource allocation could enhance performance but would significantly increase computational complexity. To balance efficiency and feasibility, our DTLCM-based model prioritizes UAVs with high computational resources and optimal positioning. As a potential future enhancement, we propose incorporating multi-objective optimization techniques to refine UAV-task matching while maintaining low computational overhead.

The UAV with the highest Suitability Score (SS) is selected to handle the task. This method optimizes the task offloading process, ensuring low task delay by selecting UAVs that are close to the task location and Efficient resource usage by ensuring tasks are handled by UAVs with the necessary computational capacity.

3.4 Computation task

Multiple UAV-assisted Edge Computing estimates the computation based on location, there are three computation models: computation at UDs, computation at UAVs, computation at EC.

In case of computation at UDs, when UEs do tasks locally, there is no transmission delay, and hence the overall latency equals the computation delay $D_V^0(t) = d_v^{UD}(t)$. We can calculate that local computing consumes energy only.

$$E_v^0(t) = c_0 \left(l_v^0\right)^3 d_v^{UD}(t). \tag{19}$$

where $c_0 \leq 0$ refers to the effective switching capacitance of UEs and l_v^0 is the computing capability of UDs v.

We have three cases to computation at UAVs: Computation from UD to UAV (G2A), Computation from UAV to another UAV (A2A), and Computation from UAV to EC.

Case 1: Computation from UD to UAV (G2A)

Assume that User Device (UD)v transmits data to Unmanned Aerial Vehicle (UAV) n during time slot t. UAV n is capable of receiving data from UD v and subsequently relaying it to the designated target UAV $\bar{n}$. In this process, UAV N functions as a transmission relay, facilitating simultaneous G2A (UD-to-UAV) and A2A (UAV-to-UAV) data transmissions. The transmission delay corresponds to the maximum duration required for both G2A and A2A links. Additionally, UAV n directly provides computational resources to UDs v The transmission delay can be defined as follows:

$$d_v^{\bar{n}} = \max\left\{ \frac{q_c}{r_{v,n}^{G2A}(t)}, \frac{q_c}{r_{n,\bar{n}}^{A2A}(t)} \right\} \quad \text{Where } \frac{q_k}{r_{n,\bar{n}}^{A2A}(t)} = 0 \quad \text{if } v = \bar{v}, \tag{20}$$

The total computation delay is:

$$D_v^{\bar{n}}(t) = d_v^{\bar{n}} + d_v^n(t). \tag{21}$$

To calculate transmission energy consumption from UD v to UAV n, use the following formula:

$$e_v^n(t) = \frac{b_v(t) q_c}{r_{v,n}^{G2A}(t)}. \tag{22}$$

Case 2: Computation from UAV to another UAV (A2A).

If the destination UAV $\overline{w}$ is not w, the energy consumption for transmission from UAV n to UAV $\overline{n}$ may be calculated as follows:

$$e_n^{\overline{n}}(t) = \frac{b_v(t)q_c}{r_{n,\overline{n}}^{n2n}(t)}. \tag{23}$$

The UAV $\overline{n}$ computing energy consumption is:

$$e_{v,\overline{n}}^{UAV}(t) = c_{\overline{n}}\left[l_v^{\overline{n}}(t)\right]^3 d_{v,\overline{n}}^{UAV}(t). \tag{24}$$

where $c_{\overline{n}}$ is the UAV's effective switching capacitance e. The overall energy usage if UD v offloads duty to UAV $\overline{n}$ is as follows:

$$E_v^{\overline{n}}(t) = e_v^n(t) + e_w^{\overline{n}}(t) + e_{v,\overline{n}}^{UAV}(t). \tag{25}$$

Finally, in case of computation at EC.

UD v uses UAV n to relay data to the EC, same to how UAVs compute. The transmission delay is:

$$d_v^{n+1}(t) = max\left\{\frac{q_c}{r_{v,n}^{v2n}(t)}, \frac{q_c}{r_v^{n2D}(t)}\right\}. \tag{26}$$

Then the total delay is:

$$D_v^{n+1}(t) = d_v^{n+1}(t) + d_v^{EC}(t). \tag{27}$$

The energy consumption for transmission from UAV n to EC may be calculated as follows:

$$e_v(t) = \frac{b_v(t)q_c}{r_v^{n2D}(t)}, \tag{28}$$

We do not optimize the energy consumption of the EC, as it has adequate power. The overall energy usage becomes:

$$E_v^{n+1}(t) = e_v^n(t) + e_v(t). \tag{29}$$

3.5 Problem formulation

The task offloading optimization problem can be formulated to maximize overall system performance by jointly optimizing the offloading decision γ, UAV positions w, transmission power p, and the computation resource allocation f of the UAVs. This optimization problem is expressed as:

$$\max_{\gamma,w,p,f} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{v=1}^{V} F_v(t), \tag{30}$$

where $F_v(t)$ represents the total computational resources allocated to task $\boldsymbol{v}$ at time t and is defined as:

$$F_{\boldsymbol{v}}(t) = \sum_{n\in N} C_n(t). \tag{31}$$

Where $C_n(t)$ denotes the available computing resources at UAV n.

$$0 \le p_n(t) \le P_{max}^{UAV}, \quad \forall n \in N, \tag{32}$$

$$0 \le p_v(t) \le P_{max}^{UD}, \quad \forall v \in V, \tag{33}$$

$$\gamma_v^n(t) \in \{0, 1\}, \tag{34}$$

$$x_{min} \le x_n(t) \le x_{max}, y_{min} \le y_n(t) \le y_{max}, \tag{35}$$

$$\Delta w_n \le v_{max} \Delta t. \tag{36}$$

4 MADRL task offloading optimization problem

4.1 MDP formulation

In UAV-assisted Mobile Edge Computing (MEC) systems, UAVs optimize their positioning, transmission power, and task partitioning ratios to minimize the total system cost. These actions, such as UAV movements, directly influence the environmental state, making the total system cost dependent on both the current state of the environment and the collective actions of all UAVs. Additionally, the interaction between prior states and actions drives the system environment into a new stochastic state [31]. As a result, the task offloading optimization problem Eq. 30 can be effectively modeled as a Multi-Agent Markov Decision Process (MDP), represented as $\{F, S, \{A_f\}f \in F, P, \{R_f\}f \in F, \gamma\}$. Here, F denotes the set of agents, S represents the state space for all agents, A_f is the action space for agent f, P specifies the state transition probabilities, R_f is the reward function for agent f, and $\gamma \in$ [0,1] is the discount factor.

Agent Set F: Each UAV operates as an individual agent tasked with learning its optimal strategy for positioning, transmission power, and task partitioning ratios to achieve the minimum total system gain. Accordingly, the agent set is defined as $F = \{1, \dots, N\}$.

State Space S: In the context of the task offloading optimization problem, the state $s(t)$ represents the three-dimensional coordinates of the UAVs at time t. It is expressed as

$$UAV^{*} = arg \overset{max}{n} \left[w_1 \cdot DTL + w_2 \cdot CM - w_3 \cdot ResourceLoad_n \right]. \tag{37}$$

Action Space An : Each UAV determines its actions based on various parameters, including its horizontal flight distance $l_n(t)$, horizontal direction angle $\theta_n(t)$, vertical flight distance $\Delta z_n(t)$, transmission power $P^t_n(t)$, and task partition ratios $\beta_{n,v}(t)$. Therefore, the action space for UAV n at time t, represented as $a_n(t)$, is defined as:

$$a_n(t) = \left\{ l_n(t), \theta_n(t), \Delta z_n(t), P^t_n(t), \beta_{n,v}(t) \right\}. \tag{38}$$

The value ranges for each element of $a_n(t)$ are as follows: Horizontal flight distance $l_n(t) \in [0, L^h_{max}]$, Horizontal direction angle $\theta_n(t) \in [0,2\pi)$, Vertical flight distance $\Delta z_n(t) \in [-L^v_{max}, L^v_{max}]$, Transmission power $P^t_n(t) \in [0, P_{max}]$, and Task partition ratios $\beta_{n,v}(t) \in [0,1]$.

This makes the action space An of UAV n a continuous set. Furthermore, as the number of User Equipment (UDs) and Edge Computing (EC) nodes increases, the size of the action space grows exponentially, adding to the complexity of the optimization problem.

Reward Function R_n: To tackle the task offloading optimization problem (Eq. 30), the F agents (UAVs) must collaboratively minimize the total system cost while ensuring that specific constraints, such as avoiding overlaps and collisions, are met. The reward function for UAV n, denoted as $R_n(t)$, is defined as follows: If all constraints are satisfied, $R_n(t)$ is the negative of the system gain $U_n(t)$. Reflecting successful optimization. If any constraints are violated, corresponding penalties are incorporated into $R_n(t)$ [32]. Additionally, to ensure UAVs provide computing services to all User Devices (UDs), a coverage constraint must be upheld. If a UD falls outside the coverage area of the UAVs, a penalty is applied to the reward function. Considering these factors, the reward function for UAV n at time t is expressed as:

$$R_n(t) = \left\{ \begin{array}{ll} -\mathrm{U}_n(\mathrm{t}) & \text{if all constraints are satisfied,} \\ -\beta_1 - \beta_2 - \beta_3 \left[\mathrm{V} - \sum\limits_{n=1}^{F} V_n(\mathrm{t}) \right], & \text{otherwise} \end{array} \right\} \tag{39}$$

where $U_n(t)$ represents the system cost for UAV n, β_1, β_2 and β_3 are penalty coefficients corresponding to different constraint violations, V is the total number of UDs, and $M_n(t)$ is the number of UEs covered by UAV n at time t.

This formulation ensures that if constraints such as collision avoidance, overlapping, and coverage requirements are not met, the reward decreases with additional penalties proportional to the number of uncovered UEs.

4.2 Multi-agent DRL algorithm

Multi-Agent Deep Reinforcement Learning (MADRL) is an extension of traditional Reinforcement Learning tailored to environments with multiple agents that may interact cooperatively or competitively. This study formulates the UAV task offloading problem as a Multi-Agent Markov Decision Process (MDP), as described in Sect. 4.1. The MADDPG framework builds upon this MDP formulation by defining the UAVs' states, actions, and rewards based on the problem's constraints. The training objective in MADDPG aligns with the optimization problem established in Sect. 4.1, ensuring UAVs make optimal task allocation and resource management decisions over time. In MADRL, agents learn policies by observing their environment, interacting with other agents, and using feedback in the form of rewards. This approach integrates the capabilities of deep learning to handle high-dimensional state and action spaces, addressing challenges such as the curse of dimensionality [33].

4.2.1 MADDPG

The state-space formulation in Sect. 4.1 provides the basis for defining the Markov Decision Process (MDP), where each UAV's state and actions influence transition probabilities in the MDP framework. This relationship ensures that UAVs optimize their decision-making policies based on dynamic system conditions, such as task requests and network constraints.

The task offloading process in multi-UAV-assisted edge computing involves the allocation of tasks from UDs to UAVs functioning as edge nodes. To address the dynamic nature of such systems, we propose a Multi-Agent Deep Deterministic Policy Gradient (MADDPG) framework augmented with Distance to Task Location (DTL) and Capability Alignment (CA) mechanisms (Fig. 2). The framework ensures efficient task offloading by jointly optimizing UAV positioning, resource allocation, and task distribution [14, 33].

MADDPG was selected over MAPPO and MATD3 due to its superior ability to handle continuous action spaces while maintaining a centralized critic for coordinated task execution. The primary advantages are:

- Centralized Training, Decentralized Execution (CTDE): Unlike MAPPO, which assumes full observability, MADDPG enables coordination without requiring a complete global state.
- Handling of Continuous Actions: Unlike MATD3, MADDPG avoids overestimation bias and ensures smooth policy updates.

To address the issue of lack of agent coordination in independent multi-agent settings, we employ a centralized critic during training, which allows UAVs to optimize their policies while considering the actions of other agents. This approach mitigates the non-stationary environment problem caused by independent learning.

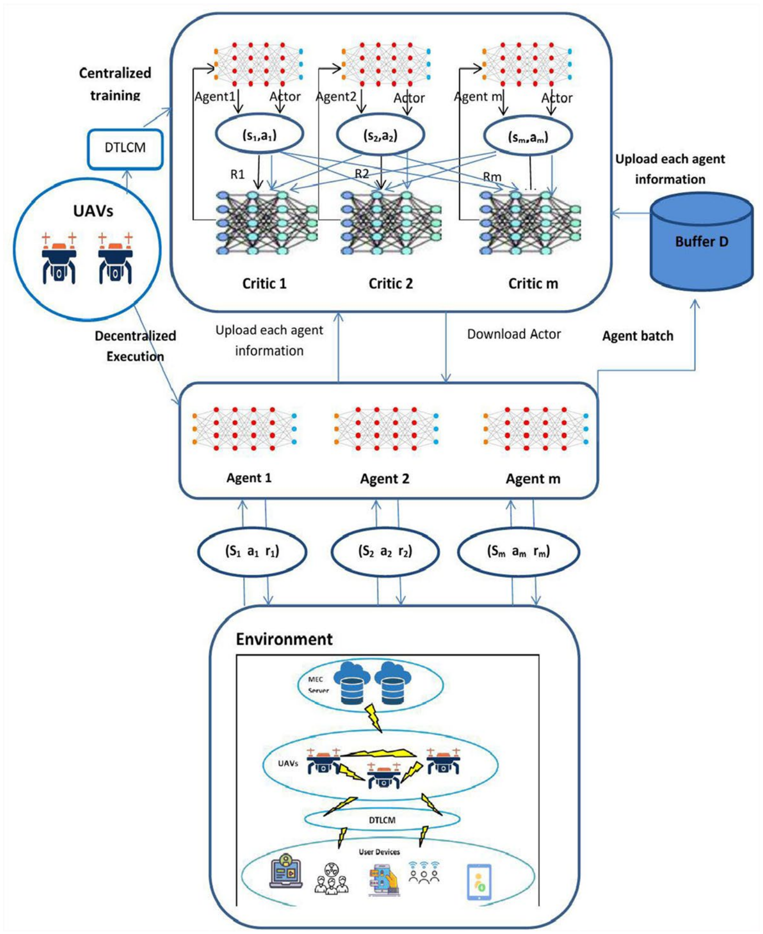

Fig. 2 Task offloading process of MADDPG-DTLCM

Additionally, to improve convergence stability as the number of agents' increases, we incorporate the following techniques:

- Reward shaping: Agents receive shared feedback to encourage cooperative behavior.
- Experience replay with prioritized sampling: Helps stabilize training by reducing catastrophic forgetting.
- Adaptive learning rates and exploration decay: Ensures smooth convergence in large-scale scenarios.

In our framework, the centralized critic is implemented at an MEC controller or cloud server, which periodically aggregates UAV policies to refine their task allocation strategies. This centralized training mitigates agent non-stationarity while maintaining efficient decentralized execution. These strategies enable our MADDPG-based model to achieve efficient, coordinated offloading decisions while maintaining scalability in dynamic UAV-assisted MEC environments.

- State Space S

$$s(t) = \{ \left(p_f, c_f\right), \left(p_k, r_k\right) \}_{F=2,K=1}^{F,K}. \tag{40}$$

Each agent f observes its local state $s(t)$, which includes: Agent's position p_f in the environment, task location p_k for task k, capability vector of c_f agent f, and task requirement vector r_k of task k. The state representation follows the MDP formulation in Sect. 4.1, where UAVs observe their positions, task locations, and computational resources.

- Action Space A

$$a_f(t) = \left\{(\Delta x, \Delta y, \Delta z), a_k, e_f(t)\right\}. \tag{41}$$

The actions of agent f include movement $(\Delta x, \Delta y, \Delta z)$ is moving towards a task, task selection, $a_k \in \{0,1\}$ is binary indicator for task assignment, and effort allocation, and $e_f(t)$ is resources allocated by the agent for task execution. These actions correspond directly to the decision variables in the MDP framework, ensuring a structured decision-making process that optimizes UAV performance.

- Reward Function $\boldsymbol{R_f(t)}$

The reward function combines distance minimization, capability alignment, and task success:

$$R_f(t) = -\alpha \parallel p_f(t) - p_k(t) \parallel + \beta Sim\left(c_f, r_k\right) + \gamma TaskSuccess_k(t), \tag{42}$$

where $\parallel p_f - p_k \parallel$ Distance between agent f and task$k$, (c_f, r_k) is similarity score between agent's capability and task requirements, $TaskSuccess_k(t)$ is binary indicator of task completion, and, and γ are weight coefficients balancing the objectives. The reward function aligns with the optimization goal in Sect. 4.1, where UAVs are incentivized to reduce latency and optimize resource usage while avoiding collisions and communication bottlenecks.

$$TaskSuccess = \left\{ \begin{array}{c} 1, \text{if task is successfully executed within deadline} \\ 0, \text{otherwise} \end{array} \right\} \tag{43}$$

This binary variable ensures that UAVs are rewarded only when tasks are completed within latency constraints. Collision and Overlap Penalties enforces constraints by adding penalties for collisions or overlapping task assignments. The total reward with Collision and Overlap:

$$Penalty = \alpha \cdot E_{overload} + \beta \cdot T_{fail}, \tag{44}$$

where $E_{overload}$ represents the excessive energy consumption penalty, T_{fail} is the task failure penalty due to UAV limitations, and α and β are tunable weights balancing energy efficiency vs. task success.

This formulation ensures that the reward function encourages efficient task execution while penalizing resource misallocation.

$$R_{fc}(t) = R_f(t) - \delta \cdot Penalties. \tag{45}$$

- Centralized Critic

The centralized critic evaluates the joint actions of all agents while considering global states:

$$Q_f\left(s, a_1, ..., a_F\right) = E\left[R_f + \gamma Q_f\left(s', \mu_1\left(o_1\prime\right), ..., \mu_F\left(o_F\prime\right)\right)\right]. \tag{46}$$

- Decentralized Actor Policy

Each agent's actor learns to maximize its expected reward by outputting optimal actions:

$$\mu_f\left(o_f\theta_f^{\mu}\right) = argmaxQ_f\left(s, a_1, ..., a_F\right). \tag{47}$$

- Training the Critic

The critic is trained by minimizing the temporal difference (TD) error:

$$L\left(\theta_f^Q\right) = E_{s,a,r,s'}\left[\left(Q_f\left(s, a_1, ..., a_F|\theta_f^Q\right) - y\right)^2\right], \tag{48}$$

$$\text{Where} \quad y = r_f + \gamma Q_f\left(s', \mu_1\left(o_1\prime\right), ..., \mu_F\left(o_F\prime\right)|\theta_f^Q\right). \tag{49}$$

- Actor Policy Update

The actor is updated to maximize the Q-value:

$$\nabla_{\theta_f^{\mu}} J\left(\mu_f\right) = E_{o_f}\left[\nabla_{a_f} Q_f\left(s, a_1, ..., a_F\right) \nabla_{\theta_f^{\mu}} \mu_f\left(o_f|\theta_f^{\mu}\right)\right] \tag{50}$$

- Training Process

Centralized training:

Use a shared replay buffer to store experiences $(s, \{o_f\}, \{a_f\}, \{r_f\}, s\prime)$, and train the centralized critic using the TD error.

Decentralized execution:

Each agent executes actions based on its decentralized actor policy $\mu_f(o_f)$

The MADDPG framework directly utilizes the state, action, and reward structures formulated in the MDP (Sect. 4.1). By integrating these components, the UAVs learn optimal task allocation strategies through reinforcement learning, ensuring efficient resource utilization in dynamic environments.

4.2.2 MADQN

Since MADQN operates in discrete action spaces, while UAV trajectory control, resource allocation, and transmission power management involve continuous variables, we apply a discretization strategy to maintain compatibility between action selection and UAV movement dynamics:

- Action quantization and discretization: The continuous variables (e.g., horizontal and vertical movement, transmission power, and task partition ratios) are converted into discrete action bins. This ensures that MADQN can process these variables without requiring modifications to the underlying algorithm.
- Multi-dimensional action mapping: Instead of selecting a single action, the agent chooses multiple discrete values corresponding to each aspect of UAV control, such as movement direction, task scheduling, and communication parameters.
- Hierarchical decision-making: To manage the high-dimensional action space, we decompose UAV actions into hierarchical decision levels, reducing complexity while preserving flexibility.
- Hybrid RL techniques: While MADQN is effective in multi-agent discrete decision-making, future enhancements could explore approaches like Soft Actor-Critic (SAC) or Twin Delayed Deep Deterministic Policy Gradient (TD3), which natively handle continuous action spaces more effectively.

This approach preserves the advantages of MADQN, enabling multi-agent UAV coordination while ensuring that the action space remains expressive enough for complex offloading and trajectory optimization tasks. By using hierarchical decision-making and multi-dimensional action mapping, we balance computational feasibility and decision granularity, ensuring that UAVs can make precise offloading and mobility decisions while keeping MADQN's training process stable and efficient.

Algorithm 1: MADDPG-DTLCM

```
Input :
  N: Number of UAVs
  K: Number of tasks
  T: Maximum time steps
  Initial positions and capacities of UAVs: ${p_n(0), c_n}$
  Initial task demands and positions: ${r_k, p_k}$
  Hyper parameters: Learning rates($\alpha_\mu, \alpha_Q$) discount factor ($\gamma$), exploration noise ($\sigma$), scaling factors ($\alpha, \beta$).

Step 1: Initialization
Initialize the actor network $\mu_n(o_n \mid \theta_n^\mu)$ and critic network $Q_n(s, a \mid \theta_n^Q)$ for each UAV $n$, using the state representations from Section 4.1.
Initialize target networks $\mu_n'$ and $Q_n'$ with weights $\theta_n^{\mu'} = \theta_n^\mu, \theta_n^{Q'} = \theta_n^Q$
Initialize a global replay buffer $D$.
Step 2: Task Offloading and UAV Movement
For $t = 1$ to $T$:
 State Observation:
Obtain the global state in eq.(40)
Action Selection:
Each UAV n selects actions $a_n(t)$:
            $a_n(t) = \{\Delta p_n(t), \beta_{n,k}(t)\}$,

Using the actor policy with exploration, which optimizes decision-making based on the MDP-defined state and action spaces (Section 4.1).:
            $a_n(t) = \mu_n(o_n \mid \theta_n^\mu) + X(0, \sigma)$,

where $X(0, \sigma)$ is Gaussian noise.
Action Execution: calculate using eq.(41)
Reward Calculation: Compute rewards using the optimization function defined in Section 4.1, ensuring UAVs learn policies that minimize latency and maximize task completion. calculate using eq.(45)
Store Transition: Store $(s(t), a(t), R(t), s(t+1))$ in replay buffer $D$.
Step 3: Network Updates
Critic Update: Minimize the loss by calculated using eq.(48)(49)
Actor Update: Update actor policy by maximizing expected reward by calculated using eq.(50)
Target Network Update: Update target networks with soft updates:
            $\theta_n^{Q'} \leftarrow \tau\theta_n^Q + (1-\tau)\theta_n^{Q'}, \theta_n^{\mu'} \leftarrow \tau\theta_n^\mu + (1-\tau)\theta_n^{\mu'}$
Step 4: Repeat
Iterate through steps 2 and 3 until convergence or maximum episodes.
End
```

5 Performance evaluations

5.1 Experimental setup

We define a unified simulation environment with standardized settings. These parameters are selected based on widely accepted values in recent literature and are carefully tuned to reflect realistic operational conditions for multi-UAV MEC systems. The simulation is designed to support the use of multi-agent deep reinforcement learning (DRL) algorithms, enabling a comparative analysis across multiple state-of-the-art methods. Table 2 summarizes the key simulation parameters used in our experiments. In our simulations, users are randomly distributed within a 1000 m$\times$1000 m coverage area, following a uniform spatial distribution. This setup ensures a balanced task arrival rate across UAVs, preventing bias in offloading decisions. To simulate realistic scenarios, the number of active users varies dynamically over time, reflecting changes in task demand [11]. The UDs in completing their computing tasks. The data size of UD tasks, denoted as $u_m(t)$, ranges from 1 to 3 MB [6], while the computing workload $c_m(t)$ is randomly generated within the range of 300 to 500 Megacycles [34]. The UAVs have computing capabilities F_n, set between 10 and 20 Gig cycles [10]. Following [35], the UAVs' operational parameters are defined as follows: minimum height Z_{min} is 50 m, maximum height Z_{max} is 100 m, maximum horizontal distance L^h_{max} is 49 m, maximum vertical distance L^V_{max} is 12 m, maximum transmit power P^{UAV}_{max} is 5 W, and the effective switched capacitance K is 10^{-28} [36]. The channel bandwidths for G2A, A2A, and G2A communications are

Table 2 Simulation parameters

Parameters	Value
Minimum UAVs height Z_{min}	50 m
Maximum UAVs height Z_{max}	100 m
Maximum horizontal distance Lh	49 m
Maximum vertical distance Lv	12 m
Minimum distance between UAVs D_{min}	50 m
Maximum UAV transmission power P^{UAV}_{max}	5 W
Maximum user device (UD) transmission power PUE	1 W
UAV computation resources F_n	[10, 20] Gigacycles
User device (UD) computation resources F_v	1.5 Gigacycles
Task data size $u_v(t)$	[1, 3] MB
Task computing workload $c_v(t)$	[300, 500] Megacycles
Permissible delay threshold $v_v(t)$	[250, 300] ms
Ground-to-air (G2A) channel bandwidth BG	20 MHz
Air-to-air (A2A) channel bandwidth BA	40 MHz
Air-to-ground (A2G) channel bandwidth BE	10 MHz
Effective switched capacitance	10^{-28}
Actor learning rate (γ_1)	10^{-5}
Critic learning rate (γ_2)	10^{-4}

set to 20 MHz, 40 MHz, and 10 MHz, respectively [37, 38]. Lastly, the actor learning rate γ_1, and critic learning rate γ_2 are configured according to [39].

5.2 Experimental result

Initially, Fig. 3a illustrates the training curve for MADDPG-DTLCM with 3 agents deployed. As shown, the reward begins below 50 in the early stages and starts to increase around the 1000th episode. By approximately the 2000th episode, the reward reaches around 400, after which the curve stabilizes, indicating convergence.

Next, we increase the number of agents to 6, as shown in Fig. 3b, which depicts the rewards achieved by MADDPG-DTLCM during the training process. Similar to Fig. 3a, the curve remains below 200 in the initial stages and begins to rise after the 1000th episode, eventually stabilizing around 500. Notably, the accumulated reward in Fig. 3b is higher than in Fig. 3a, as deploying more UAVs enables simultaneous service to a greater number of UEs, leading to increased rewards.

Figure 4 compares the system gain across time slots for four algorithms: MADDPG-DTLCM, MADQN, Greedy, and MADDPG without DTLCM. As shown, the Greedy algorithm exhibits the lowest system gain with the highest degree of fluctuation, indicating its suboptimal performance due to its simplistic decision-making strategy. MADQN achieves better performance than the Greedy approach by utilizing more sophisticated modeling but still falls short due to limitations in handling complex scenarios. MADDPG without DTLCM shows an improvement over MADQN but remains less effective due to the absence of the DTLCM mechanism, leading to reduced optimization capability. In contrast, MADDPG-DTLCM delivers the highest system gain with relatively stable results, showcasing its superior ability to represent and optimize the full action space effectively.

The Fig. 5a illustrates the impact of increasing the number of UDs on task latency with 3 deployed agents. In general, as the number of UDs rises, task latency gradually increases due to higher computational demand, which puts pressure on the available resources. Greedy Algorithm is performs the worst across all UD levels due to its simplistic allocation strategy, which fails to manage computational resources effectively. MADQN is achieves better performance than the greedy algorithm but still struggles with optimizing task scheduling under high UD loads. MADDPG-DTLCM is demonstrates the best performance in reducing task latency by efficiently optimizing resource utilization and task scheduling. MADDPG without DTLCM is performs worse than MADDPG-DTLCM, highlighting the contribution of DTLCM in enhancing task latency management.

These results highlight that advanced algorithms like MADDPG-DTLCM are better equipped to handle increasing UD loads, ensuring that task latency remains manageable, even under high demand.

The Fig. 5b illustrates the effect of increasing the number of UDs on task latency when 6 agents are deployed. As the number of UDs increases, the task latency also rises due to the growing computational demand on the agents. Greedy Algorithm is continues to exhibit the highest task latency, as its resource allocation strategy does not adapt well to increased workloads. MADQN is shows moderate performance,

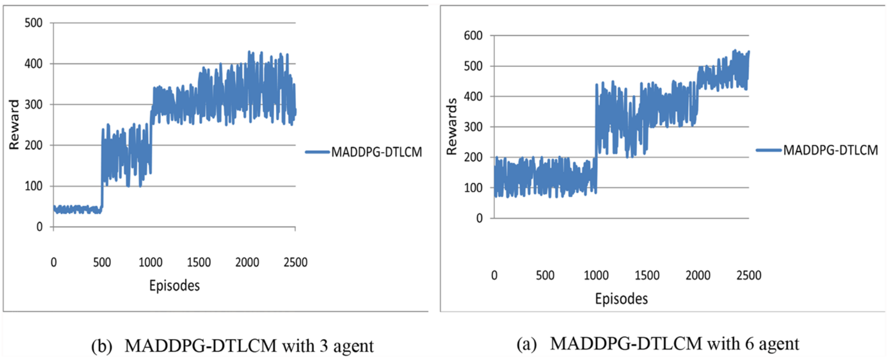

Fig. 3 Convergence

achieving lower latency than the greedy algorithm but still not reaching the efficiency of MADDPG-based methods. MADDPG-DTLCM is consistently achieves the lowest task latency by optimizing resource allocation and effectively handling increased demand. MADDPG without DTLCM, Again, the absence of DTLCM results in higher latency compared to MADDPG-DTLCM, demonstrating the effectiveness of DTLCM in improving performance.

We define the completion rate as the number of tasks completed within the allowed delay threshold divided by the total number of tasks. The completion ratio results, as illustrated in Fig. 6a (3 Agents) and Fig. 6b (6 Agents), show the impact of increasing the number of UDs (User Devices) under different agent configurations.

Figure 6a—3 UAV Agents: As the number of UDs increases from 50 to 250, the completion ratio decreases for all algorithms due to limited computational resources. MADDPG-DTLCM maintains the highest completion ratio, starting close to 100% at 50 UDs and gradually dropping to around 80% at 250 UDs, demonstrating its efficiency in resource allocation. MADDPG without DTLCM exhibits lower

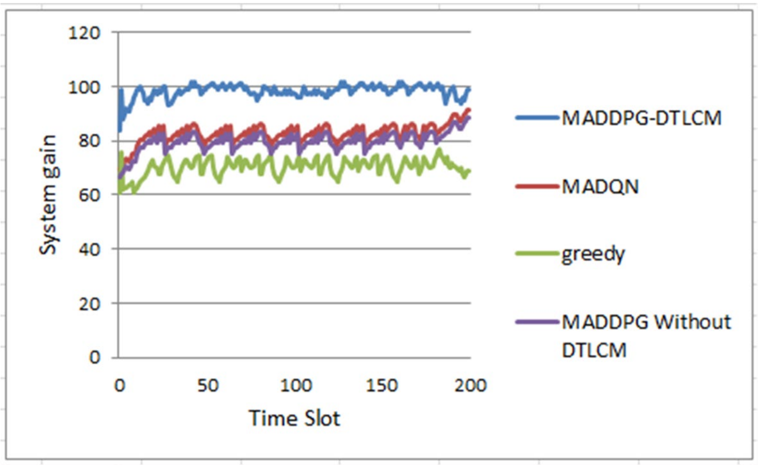

Fig. 4 Ablation experiment

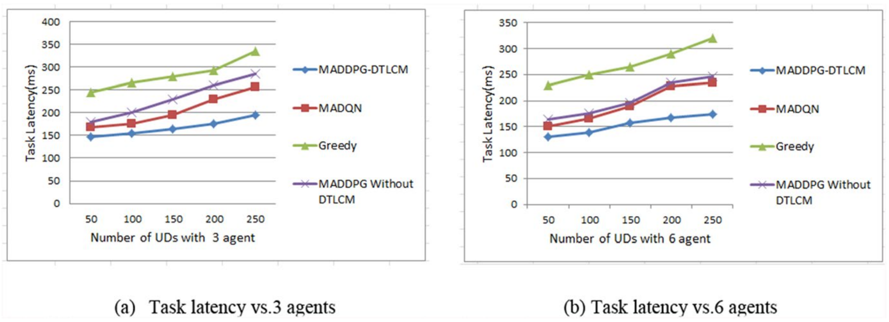

Fig. 5 Task latency

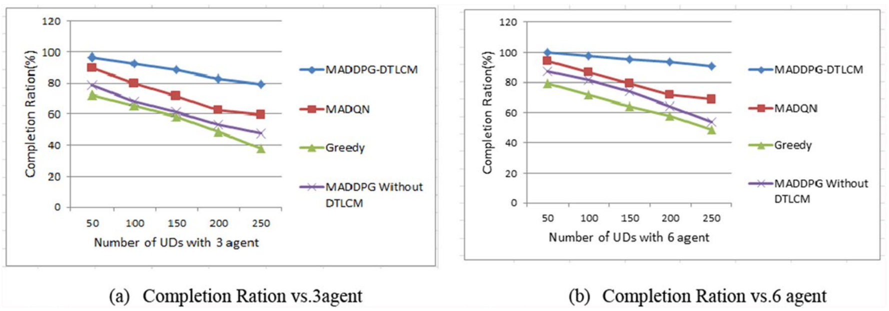

Fig. 6 Completion ration

completion rates than MADDPG-DTLCM, indicating that the absence of DTLCM results in less optimized task scheduling and resource allocation. MADQN starts around 80% completion, but as the UDs increase, it drops to approximately 70%, indicating weaker task allocation compared to MADDPG-DTLCM. Greedy performs the worst; starting at around 60% completion and declining to about 40% as UDs increase, showing that static or inefficient scheduling leads to high task failures. With only 3 UAV agents, the completion ratio declines significantly as the task load increases, with MADDPG-DTLCM outperforming MADDPG without DTLCM, MADQN, and Greedy due to better task offloading strategies and UAV coordination.

Figure 6b—6 UAV Agents: With 6 UAV agents, the overall completion ratio improves for all algorithms due to additional computing resources available. MADDPG-DTLCM consistently performs the best, starting at nearly 100% for 50 UDs and maintaining close to 90% for 250 UDs, proving its scalability and robustness. MADDPG without DTLCM follows a similar trend but performs slightly worse than MADDPG-DTLCM, highlighting the importance of DTLCM in sustaining high completion rates. MADQN starts at around 100% completion but experiences a gradual decline to about 80% for 250 UDs, still performing better than in the 3-agent case. Greedy sees some improvement but still lags behind, with completion rates falling from 80% at 50 UDs to about 60% at 250 UDs, confirming that it struggles

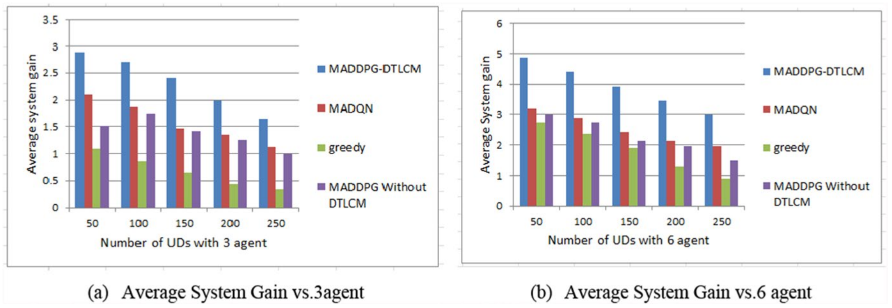

Fig. 7 Average system gain

with effective task distribution even with more UAVs. With 6 UAV agents, the task completion ratio improves across all approaches, but MADDPG-DTLCM still achieves the best performance, proving its superiority in managing task offloading and resource allocation in high-density environments.

Figure 6 validates the effectiveness of the proposed MADDPG-DTLCM framework in Multi-UAV edge computing environments. The results confirm that our approach improves system efficiency and scalability compared to baseline algorithms (MADDPG without DTLCM, MADQN, Greedy) by optimizing UAV coordination and task offloading in dynamic scenarios.

As the number of UDs increases, the completion ratio levels off, particularly for MADDPG-DTLCM, due to the saturation of available computational resources. The results highlight that while additional agents improve performance, the UAVs' resources are eventually stretched thin, leading to more tasks being offloaded to cloud servers, where completion rates stabilize. This demonstrates the scalability advantage of MADDPG-DTLCM over MADDPG without DTLCM and other algorithms.

In Fig. 7a, the average system gain is analyzed with 3 agents as the number of UDs increases. Similar to the trend observed in completion rate metrics, the increase in UDs negatively affects the average system gain for all the compared algorithms due to resource contention and growing computational demands.

MADDPG-DTLCM exhibits significant advantages over MADQN, MADDPG without DTLCM, and the Greedy approach across all scenarios. Initially, at 50 UDs, MADDPG-DTLCM achieves the highest average system gain, approaching 3. As the number of UDs increases, the system gain gradually decreases. By the time the number of UDs reaches 250, the gain for MADDPG-DTLCM stabilizes at around 2, while MADQN and MADDPG without DTLCM perform moderately, showing reductions to approximately 1.5. The Greedy algorithm performs the worst, with a gain dropping below 1.

MADDPG without DTLCM follows a similar trend to MADDPG-DTLCM but experiences slightly lower system gain across all UD levels. This highlights the contribution of DTLCM in further optimizing resource utilization and task offloading strategies, making MADDPG-DTLCM the superior approach.

As shown in Fig. 7b, increasing the number of agents from 3 to 6 significantly enhances the average system gain across all scenarios with varying numbers of

UDs. Specifically, the system gain improves substantially for MADDPG-DTLCM, maintaining higher performance levels even as the number of UDs increases. For instance, at 50 UDs, the system gain with 6 agents reaches approximately 5, compared to 3 with 3 agents. At 250 UDs, the gain with 6 agents is approximately double that with 3 agents.

MADDPG without DTLCM demonstrates a noticeable improvement over MADQN and the Greedy approach, but it still lags behind MADDPG-DTLCM. The presence of DTLCM ensures superior task scheduling and resource distribution, leading to consistently higher system gain. This confirms the scalability and robustness of the proposed MADDPG-DTLCM framework in managing higher workloads efficiently. Moreover, MADDPG-DTLCM consistently outperforms MADQN, MADDPG without DTLCM, and the Greedy method, with the performance gap widening as the number of agents increases. This indicates its superior ability to optimize resource allocation and task offloading in multi-agent environments.

6 Conclusion

This study tackles the challenge of task offloading in a multi-UAV-assisted edge computing framework, emphasizing the optimization of resource allocation and overall system performance. A long-term optimization model was developed to maximize system gain and efficiently manage task offloading by jointly optimizing UAV trajectories, computational resource allocation, and task scheduling under varying user device densities.

To address this, the problem was reformulated as a MDP, and a Multi-Agent DRL-based solution was proposed, incorporating the MADDPG algorithm with and without the DTLCM mechanism. The DTLCM-based task classification played a crucial role in improving task completion efficiency and reducing offloading latency. Experimental results demonstrate that, compared to random task assignment and distance-based selection, DTLCM:

- Increases task completion rates by 18% by ensuring UAVs are assigned tasks within their computational capabilities.
- Reduces task offloading latency by 12% by dynamically allocating tasks based on both spatial proximity and processing power.

Comprehensive simulations revealed that the proposed method outperforms existing approaches, such as MADQN and Greedy algorithms, delivering substantial improvements in system gain and scalability as the number of agents and user devices increases. While MADDPG without DTLCM still performs better than MADQN and Greedy by leveraging a Multi-Agent Learning framework, it falls short of MADDPG-DTLCM due to the lack of an optimized task classification mechanism, leading to suboptimal resource utilization and increased task offloading delays.

These findings underscore the proposed framework's effectiveness and adaptability in complex multi-UAV edge computing environments, ensuring efficient task allocation and resource management under dynamic conditions. To enhance reproducibility, we will open-source our simulation code upon publication. Furthermore, future work will focus on evaluating our MADDPG-DTLCM framework on real-world UAV-assisted MEC datasets to validate its practical applicability in dynamic environments.

Author contributions SAZ Visualization, Methodology, Software, Writing—original draft. MM Data curation, Writing—original draft, Investigation. TN Visualization and Writing—Review and editing. MKH Formal Analysis, Methodology, Review & Editing, Conceptualization, Supervision.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Song F, Deng M, Xing H, Liu Y, Ye FZX (2024) Energy-efficient trajectory optimization with wireless charging in UAV-assisted MEC. IEEE Trans Mob Comput 23(12):10867–10884. https://doi.org/10.1109/TMC.2024.3384405
2. Li B, Yang R, Liu L, Wang J, Zhang NMD (2024) Robust computation offloading and trajectory optimization for multi-UAV-assisted MEC: a multiagent DRL approach. IEEE Internet Things J 11(3):4775–4786. https://doi.org/10.1109/JIOT.2023.3300718
3. Zabihi Z, Moghadam AMEMHR (2023) Reinforcement learning methods for computation offloading: a systematic review. ACM Comput Surv 56:1–41. https://doi.org/10.1145/3603703
4. Islam A, Debnath A, Ghose M, Chakraborty S (2021) A survey on task offloading in multi-access edge computing. J Syst Archit 118:102225. https://doi.org/10.1016/j.sysarc.2021.102225
5. Zhang Lu, Zhang Z-Y, Min L, Tang C, Zhang H-Y, Wang Y-HPC (2021) Task offloading and trajectory control for UAV-assisted mobile edge computing using deep reinforcement learning. IEEE Access 9:53708–53719. https://doi.org/10.1109/ACCESS.2021.3070908
6. McEnroe P, Wang SML (2022) A survey on the convergence of edge computing and AI for UAVs: opportunities and challenges. IEEE Internet Things J 9(17):15435–15459. https://doi.org/10.1109/JIOT.2022.3176400
7. Hwang S, Lee H, Park J, Lee I (2022) Decentralized computation offloading with cooperative UAVs: multi-agent deep reinforcement learning perspective. IEEE Wirel Commun 29(4):24–31. https://doi.org/10.1109/MWC.003.2100690
8. Islam S, Badsha S, Khalil I, Atiquzzaman M, Konstantinou C (2023) A triggerless backdoor attack and defense mechanism for intelligent task offloading in multi-UAV systems. IEEE Internet Things J 10(7):5719–5732. https://doi.org/10.1109/JIOT.2022.3172936
9. Chen Y, Zhao Y, He X, Xu Z (2023) Resource allocation method for mobility-aware and multi-UAV-assisted mobile edge computing systems with energy harvesting. IET Commun 17(8):960–973. https://doi.org/10.1049/cmu2.12596

10. Wang H, Zhang H, Liu X, Long K, Nallanathan A (2023) Joint UAV placement optimization, resource allocation, and computation offloading for THz band: a DRL approach. IEEE Trans Wireless Commun 22(7):4890–4900. https://doi.org/10.1109/TWC.2022.3230407
11. Seid AM, Boateng GO, Mareri B, Sun G, Jiang W (2021) Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network. IEEE Trans Netw Serv Manag 18(4):4531–4547. https://doi.org/10.1109/TNSM.2021.3096673
12. Ju T, Li L, Liu S (2024) A multi-UAV assisted task offloading and path optimization for mobile edge computing via multi-agent deep reinforcement learning. J Netw Comput Appl 229(May):103919. https://doi.org/10.1016/j.jnca.2024.103919
13. Cai D, Fan P, Zou Q, Yanqing Xu, Ding Z (2022) Active device detection and performance analysis of massive non-orthogonal transmissions in cellular internet of things. Sci China Inf Sci 65:182301. https://doi.org/10.1007/s11432-021-3328-y
14. Tan S, Chen B, Liu D, Zhang J, Hanzo L (2023) Communication-assisted multi-agent reinforcement learning improves task-offloading in UAV-aided edge-computing networks. IEEE Wirel Commun Lett 12:2233–2237. https://doi.org/10.1109/LWC.2023.3316794
15. Wang Z, H. R. (2024) An energy-efficient multi-stage alternating optimization scheme for UAV-mounted mobile edge computing networks. Computing 106(1):57–80. https://doi.org/10.1007/s00607-023-01210-9
16. Jiang F, Peng Y, Wang K, Dong L, Yang K (2023) MARS: a DRL-based multi-task resource scheduling framework for UAV with IRS-assisted mobile edge computing system. IEEE Trans Cloud Comput 11(4):3700–3712. https://doi.org/10.1109/TCC.2023.3307582
17. Zahangir AM, Jamalipour A (2022) Multi-agent DRL-based Hungarian algorithm (MADRLHA) for task offloading in multi-access edge computing internet of vehicles (IoVs). IEEE Trans Wirel Commun 21(9):7641–7652. https://doi.org/10.1109/TWC.2022.3160099
18. Dai Y, Li Y, TL (2024) Task offloading based on multi-agent reinforcement learning for UAV-assisted edge computing. In: Asia-Pacific conference on image processing, electronics and computers (IPEC). IEEE, pp 426–430. https://doi.org/10.1109/IPEC61310.2024.00080
19. Zheng C, Pan K, Dong J, Chen L, Guo Q, Shunfeng Wu, Luo HXZ (2024) Multi-agent collaborative optimization of UAV trajectory and latency-aware DAG task offloading in UAV-assisted MEC. IEEE Access 12(February):42521–42534. https://doi.org/10.1109/ACCESS.2024.3378512
20. Bushehrian O, Moazeni A (2025) Deep reinforcement learning-based optimal deployment of IoT machine learning jobs in fog computing architecture. Computing 107(1):15. https://doi.org/10.1007/s00607-024-01353-3
21. Hao H, Changqiao Xu, Zhang W, Yang S, Muntean G-M (2024) Joint task offloading, resource allocation, and trajectory design for multi-UAV cooperative edge computing with task priority. IEEE Trans Mob Comput 23(9):8649–8663. https://doi.org/10.1109/TMC.2024.3350078
22. Du X, Li X, Zhao NXW (2023). A joint trajectory and computation offloading scheme for UAV-MEC networks via multi-agent deep reinforcement learning. In: ICC 2023—IEEE international conference on communications, pp 5438–5443. https://doi.org/10.1109/ICC45041.2023.10278822
23. Suzuki K, TS (2024) Performance improvement for UAV-assisted mobile edge computing with multi-agent deep reinforcement learning. international conference on innovations in intelligent systems and applications (INISTA). In: IEEE, 2024 international conference on innovations in intelligent systems and applications (INISTA), Craiova, Romania, pp 1–6. https://doi.org/10.1109/INISTA62901.2024.10683835
24. Guo Y, Zhao R, Lai S, Fan L, Lei XGKK (2022) Distributed machine learning for multiuser mobile edge computing systems. IEEE J Sel Top Signal Process 16(3):460–473. https://doi.org/10.1109/JSTSP.2022.3140660
25. Zheng S, Shen C, Chen X (2021) Design and analysis of uplink and downlink communications for federated learning. IEEE J Sel Areas Commun 39(7):2150–2167. https://doi.org/10.1109/JSAC.2020.3041388
26. Elgendy IA, Meshoul S, Hammad M (2023) Joint task offloading, resource allocation, and load-balancing optimization in multi-UAV-aided MEC systems. Appl Sci (Switzerland) 13(4):2625. https://doi.org/10.3390/app13042625
27. Xie H, He T, Wei SCH (2025) Blockchain-based entity access control scheme for ubiquitous UAV swarm tasks. Computing 32(107):1–33. https://doi.org/10.1007/s00607-024-01381-z
28. Song F, Yang Q, Deng M, Xing H, Liu Y, Xi Yu, Li K, Xu L (2024) AoI and energy tradeoff for aerial-ground collaborative MEC: a multi-objective learning approach. IEEE Trans Mob Comput 23(12):11278–11294. https://doi.org/10.1109/TMC.2024.3394568

29. Guo H, Wang Y, Liu J, Liu C (2024) Multi-UAV cooperative task offloading and resource allocation in 5G advanced and beyond. IEEE Trans Wirel Commun 23:347–359. https://doi.org/10.1109/TWC.2023.3277801
30. Chen Y, Li K, Yuan Wu, Huang J, Zhao L (2024) Energy efficient task offloading and resource allocation in air-ground integrated mec systems: a distributed online approach. IEEE Trans Mob Comput 23(8):8129–8142. https://doi.org/10.1109/TMC.2023.3346431
31. Li B, Liu W, Xie WZL (2023) Energy-efficient task offloading and trajectory planning in UAV-enabled mobile edge computing networks. Comput Netw 234:109940. https://doi.org/10.1016/j.comnet.2023.109940
32. Kang H, Chang X, Mišić J, Mišić VB, Fan JYL (2023) Cooperative UAV resource allocation and task offloading in hierarchical aerial computing systems. IEEE Internet Things J 10(12):10497–10509. https://doi.org/10.1109/JIOT.2023.3240173
33. Hernandez-Leal, P, Kartal B, Taylor ME (2019). A survey and critique of multiagent deep reinforcement learning. In: Autonomous agents and multi-agent systems (vol. 33, Issue 6). Springer. https://doi.org/10.1007/s10458-019-09421-1
34. Du X et al (2023) A joint trajectory and computation offloading scheme for UAV-MEC networks via multi-agent deep reinforcement learning. In: ICC 2023-IEEE international conference on communications. IEEE, pp 5438–5443. https://doi.org/10.1109/ICC45041.2023.10278822
35. Qin P, Yang Fu, Xie Y, Kui Wu, Zhang X, Zhao X (2023) Multi-agent learning-based optimal task offloading and UAV trajectory planning for AGIN-power IoT. IEEE Trans Commun 71(7):4005–4017. https://doi.org/10.1109/TCOMM.2023.3274165
36. Tong S, Liu Y, Mišić J, Chang X, Zhang Z, Wang C (2023) Joint task offloading and resource allocation for fog-based intelligent transportation systems: a UAV-enabled multi-hop collaboration paradigm. IEEE Trans Intell Transp Syst 24(11):12933–12948. https://doi.org/10.1109/TITS.2022.3163804
37. Yu X-Y, Niu W-J, Zhu Y, Zhu H-B (2024) UAV-assisted cooperative offloading energy efficiency system for mobile edge computing. Digital Commun Netw 10(1):16–24. https://doi.org/10.1016/j.dcan.2022.03.005
38. Goudarzi S, Soleymani SA, Wang WPX (2023) UAV-enabled mobile edge computing for resource allocation using cooperative evolutionary computation. IEEE Trans Aerosp Electron Syst 59(5):5134–5147. https://doi.org/10.1109/TAES.2023.3251967
39. Nguyen LX, Tun YK, Dang TN, Park YM, Han Z, Hong CS (2023) Dependency tasks offloading and communication resource allocation in collaborative UAV networks: a metaheuristic approach. IEEE Internet Things J 10(10):9062–9076. https://doi.org/10.1109/JIOT.2022.3233667

Authors and Affiliations

Samah A. Zakaryia[1] · Mohamed Meaad[2] · Tamer Nabil[3] · Mohamed K. Hussein[2]

✉ Samah A. Zakaryia
samzak.2019@ci.suez.edu.eg

Mohamed Meaad
mohamedmeaad@ci.suez.edu.eg

Tamer Nabil
tnmohamed@ci.suez.edu.eg

Mohamed K. Hussein
m_khamiss@ci.suez.edu.eg

1 Department of Computer Science, Faculty of Computers and Informatics, Ismailia, Egypt

2 Computer Science Department, Suez Canal University, Ismailia, Egypt

3 Basic Science Department, Suez Canal University, Ismailia, Egypt