Journal of Systems Architecture 176 (2026) 103830

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

UAV-USV collaborative task offloading for edge computing enabled smart lake monitoring

Chaogang Tang [a], Tiyu Yao [a], Shuo Xiao [a,*], Huaming Wu [b], Ruidong Li [c]

[a] School of Computer Sciences and Technology/School of Artificial Intelligence, China University of Mining and Technology, Xuzhou, 221100, Jiangsu, China
[b] The Center for Applied Mathematics, KL-AAGDM, Tianjin University, 300072, Tianjin, China
[c] The Institute of Science and Engineering, Kanazawa University, Kanazawa, 920-1192, Japan

ARTICLE INFO

Keywords:
Mobile edge computing
Dynamic task allocation
Deep reinforcement learning
Resource allocation
Trajectory planning

ABSTRACT

Cloud computing based aquatic monitoring systems fails to achieve significant performance improvement due to the long communication latency in reality. To overcome this limitation, we propose a UAV-USV cooperative Mobile Edge Computing (MEC) framework for smart city lake monitoring. In this framework, Unmanned Surface Vehicles (USVs) are intelligently dispatched to dynamically generated suspicious task points. Unmanned Aerial Vehicle (UAV) on the other hand serves as the edge node to provide aerial computational support for tasks generated by USVs. The UAV-USV cooperation enables real-time end-to-end suspicious points monitoring and sensing task processing during a single time slot. We jointly optimize the allocation of suspicious task points, the task offloading strategy of USVs, UAV trajectory planning, and computational resource allocation in dynamic task assignment scenarios. Our objective is to maximize UAV coverage while minimizing overall system delay and energy consumption. The optimization problem is decomposed into two subproblems: suspicious task point allocation and task offloading-trajectory optimization. The former employs a greedy algorithm to achieve efficient USV scheduling, while the latter integrates Deep Reinforcement Learning (DRL) and convex optimization to jointly optimize offloading decisions, UAV trajectory, and resource allocation. Simulation results demonstrate that the proposed framework achieves superior performance in terms of system delay, energy efficiency, and task coverage, validating its feasibility and application potential in dynamic smart lake environments.

1. Introduction

With the rapid pace of urbanization, aquatic environments are facing complex challenges such as eutrophication, pollution diffusion, and ecosystem degradation. Hence, the ecological monitoring of urban water bodies, such as lakes, reservoirs, and rivers, is becoming more important, with respect to sustainable environmental management, ecological security, and intelligent ecosystem governance [1–3]. Traditional aquatic monitoring systems typically depend on fixed sensor nodes or predefined periodic task scheduling mechanisms. These systems lack flexibility and adaptability in dynamic environments, making it difficult for them to respond promptly to sudden environmental changes or emergency events [4]. In addition, the existing cloud-based centralized architectures can offer robust computational and storage capabilities. However, they are constrained by communication bandwidth and latency, consequently being unable to support real-time data processing and rapid decision-making in large-scale or remote aquatic scenarios [5].

Recently, the mobile edge computing (MEC)-based paradigm has shown promising prospects for intelligent aquatic environment monitoring. This paradigm integrates the advantages of both USVs and UAVs, helping reduce communication latency, alleviate network congestion, and minimize energy consumption [6–8]. This architecture fully leverages the complementary advantages of aerial and surface platforms. On the one hand, UAVs possess high mobility, wide communication coverage, and rapid deployment capability; on the other hand, USVs have strong endurance and high operational stability. For instance, UAVs can function as mobile edge servers to support USVs in communication and computation tasks, especially in regions where network connectivity is unreliable or terrestrial infrastructure is scarce. Compared with traditional static sensing systems, this MEC-enhanced cooperative architecture demonstrates superior advantages in dynamic water environments, in terms of adaptability, response latency, scalability, and is well-suited for next-generation intelligent water environment monitoring.

* Corresponding author.
E-mail address: sxiao@cumt.edu.cn (S. Xiao).

https://doi.org/10.1016/j.sysarc.2026.103830
Received 8 December 2025; Received in revised form 28 March 2026; Accepted 1 May 2026
Available online 7 May 2026

However, most existing studies assume static task models, meaning that the locations and quantities of tasks are predefined. Consequently, they neglect the dynamic and stochastic nature of aquatic monitoring tasks. In reality, suspicious events or abnormal monitoring points may irregularly emerge over time and space in lake environments, which requires the system to have dynamic task-processing and real-time adaptation capabilities. Worse still, jointly optimizing UAV trajectory planning, task offloading, and resource allocation under multiple system constraints is highly complex. The objective of this optimization includes maximizing UAV coverage while minimizing processing delay and energy consumption, making the problem particularly challenging.

To address the above issues, we propose a USV–UAV cooperative edge computing system for intelligent aquatic monitoring. In this system, multiple USVs autonomously cruise on the lake surface and travel to dynamically generated suspicious task points for data collection and task execution. Meanwhile a UAV equipped with a MEC module provides real-time computational support to the USVs. The UAV's position is dynamically adjusted based on the distribution and communication status of the USVs to maintain efficient coverage and computational performance. During this process, the dynamic allocation of USVs to different suspicious task points is crucial, as it can significantly impact system evaluation metrics such as task load balance, fairness, and energy consumption. Inappropriate allocation may even lead to an increase in the task process latency, which violates the real-time latency requirement.

Accordingly, this paper designs a collaborative optimization mechanism, considering task load balancing among USVs and additional energy consumption, to prevent individual USVs from being overloaded and reduce overall system energy costs. To this end, a joint optimization of suspicious task point allocation, UAV flight trajectory, and USV task offloading in a USV-UAV cooperative aquatic monitoring scenario is proposed, aiming to simultaneously minimize load-balancing-aware energy consumption, maximize UAV coverage, and minimize task response latency. The main contributions of this paper are summarized as follows:

- A USV-UAV cooperative monitoring framework is introduced, incorporating a dynamic task generation mechanism that abandons the traditional static task assumption. Unlike prior studies assuming static task sets, this mechanism adapts to real-time monitoring needs, enabling efficient task allocation with load balancing among USVs and making the system more consistent with real-world aquatic monitoring scenarios.
- A joint optimization problem is formulated, and a load-balancing-aware scheduling strategy is proposed for the allocation of suspicious task points, which also considers additional energy consumption in the optimization objective. By jointly optimizing UAV trajectory and USV offloading decisions, the approach prevents individual USVs from being overloaded while maintaining high communication coverage, providing a more balanced and practical solution compared to standard energy-minimization or greedy scheduling strategies.
- An efficient hybrid algorithm is designed to address the formulated optimization problem. Specifically, a greedy algorithm is used for dynamic task allocation among USVs considering both energy consumption and load balancing, a DRL-based algorithm is applied for UAV trajectory planning and offloading decisions, and convex optimization is employed for edge resource allocation. This combination of greedy, DRL, and convex optimization jointly addresses energy efficiency, coverage, and load balancing in a way that prior hybrid approaches combining only two techniques cannot.
- Extensive simulations are conducted under various scenarios, demonstrating that the proposed strategy consistently outperforms baseline methods across multiple performance metrics, validating both the novelty and practical effectiveness of the approach in dynamic multi-USV-UAV aquatic monitoring scenarios.

The remainder of this paper is organized as follows: Section 2 reviews recent advances in USV-based aquatic sensing and UAV-assisted MEC offloading optimization. Section 3 introduces the system model. Section 4 formulates the optimization problem. Section 5 details the proposed algorithmic framework. Section 6 presents and analyzes the simulation results, and Section 7 concludes the paper.

2. Related work

2.1. USV-based aquatic sensing

With the growing demand for intelligent aquatic monitoring and ecological protection, USVs have become key platforms for environmental sensing and data collection in aquatic environments [9,10]. Compared with fixed monitoring stations, USVs offer high mobility, wide coverage, and redeployability, enabling flexible operations in complex aquatic conditions. As multi-USV cooperative monitoring tasks grow more complex, trajectory planning and energy consumption control have emerged as key factors affecting system performance.

Toyomoto et al. [11] proposed a multi-USV cooperative monitoring method combining linear and circular trajectory planning, and tried to achieve a balance between coverage and path continuity. Oshima et al. [12] developed an online path generation algorithm based on constrained control, enabling real-time trajectory updates and dynamic cooperative coverage among multiple USVs. Zhao et al. [13] introduced a two-layer navigation strategy integrating global path planning and local waypoint control, ensuring precise tracking and strong energy efficiency even in complex environments.

In terms of energy management, Huang et al. [14] proposed a self-powered mechanism based on nonlinear ocean energy harvesting. By constructing a wave–energy conversion dynamic model, this method efficiently recycles environmental energy, enhancing USV endurance and operational stability. Additionally, Su et al. [15] proposed an intelligent trajectory planning method based on DRL, formulating the task as a Constrained Markov Decision Process and integrating target-oriented steering optimization with the Twin Delayed Deep Q-Network algorithm to jointly optimize trajectory and speed, effectively reducing energy consumption and improving communication reliability.

2.2. RL-based UAV trajectory and MEC offloading optimization

As a key technology for alleviating the computational burden on end devices, MEC has been widely applied in various unmanned systems [16,17]. In recent years, researchers have increasingly introduced UAVs into the MEC framework, enabling them to serve as aerial edge nodes that participate in computation and communication processes, thereby achieving lower latency and more efficient task processing [18,19].

To address latency issues arising from cloud-based task offloading, Wang et al. [20] introduced a UAV-assisted MEC framework that jointly optimizes task offloading, UAV trajectory, and occlusion-aware communication. The authors modeled the problem as an MDP and developed an improved adaptive delayed DDPG (AD3PG) algorithm, which dynamically adjusts connectivity, noise power, and offloading ratios. Simulation results show that AD3PG outperforms conventional DDPG and TD3 in dynamic scenarios, reducing overall system latency by up to 35.3%.

Tong et al. [21] investigated a multi-base-station collaborative mobile edge computing (MEC) system, focusing on the joint optimization of task offloading and resource pricing. In their model, a macro base station cooperates with multiple energy-harvesting-enabled micro base stations to provide computation services for dense end-users. The interactions between base stations and users are formulated as a Stackelberg game, where base stations act as leaders to set pricing strategies, and users, as followers, determine offloading decisions to

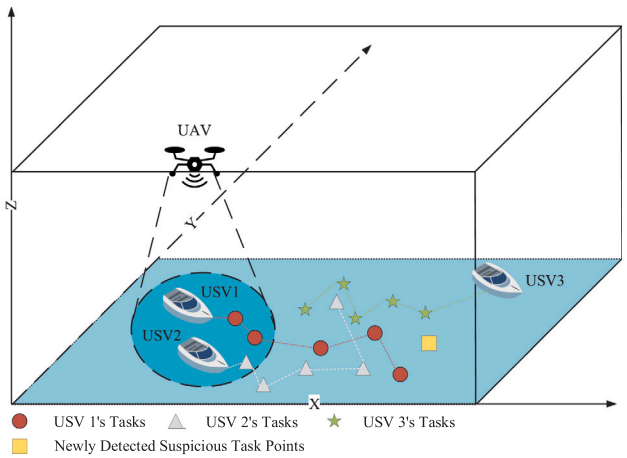


Fig. 1. Intelligent water monitoring system.

minimize their costs. To address the resulting coupled and incomplete-information problem, the authors proposed a TO-SG-MADDPG algorithm that integrates Stackelberg game theory with a multi-agent deep deterministic policy gradient framework. Through centralized training and distributed execution, the method jointly optimizes pricing and offloading policies. Simulation results show that the proposed approach effectively reduces average task latency and task loss while achieving a balanced trade-off between the utilities of service providers and users.

Zhao et al. [22] proposed a collaborative MEC framework where multiple UAVs cooperatively process offloaded tasks from mobile users, incorporate charging stations to ensure UAV endurance, and jointly optimize computation offloading, resource allocation, and charging scheduling using a TD3-based reinforcement learning algorithm, which demonstrates superior performance in terms of system utility and resource utilization compared with conventional approaches.

Wang et al. [23] proposed an intelligent algorithm integrating MADDPG, LQR, and CVXPY, achieving real-time joint optimization of UAV trajectory control and resource allocation to minimize energy consumption and task latency. Simulations demonstrate its effectiveness in complex multi-UAV cooperative edge computing scenarios.

Although research on USV-based aquatic sensing and UAV-assisted edge computing has made progress, several gaps remain. First, most existing studies rely on static or quasi-static task sets and lack mechanisms for dynamically generated suspicious tasks. Second, current task allocation and offloading strategies primarily optimize energy consumption or delay, with limited consideration of load balancing among USVs. Third, existing hybrid optimization frameworks typically combine only two types of techniques (e.g., greedy methods with convex optimization or deep reinforcement learning with convex optimization), making it difficult to jointly address dynamic task allocation, UAV trajectory control, and edge resource management. In contrast, the proposed approach integrates greedy scheduling, deep reinforcement learning, and convex optimization into a unified framework that simultaneously considers energy efficiency, coverage, and load balancing. Moreover, Yuan et al. [24] proposed AIRA, which integrates multi-agent reinforcement learning with smart contracts to jointly optimize resource allocation, task compression, and offloading with adaptive incentives. Meanwhile, Yuan et al. [25] developed JORA, which leverages blockchain-based incentive mechanisms and formulates joint offloading and resource allocation as an optimization problem solved via decomposition techniques. Although these studies address joint optimization in edge systems, our work focuses on USV–UAV cooperative aquatic monitoring, which involves dynamic task generation, UAV trajectory planning, and real-time resource allocation in maritime environments, highlighting the novelty of our approach.

To address the limitations of existing approaches, we propose a joint optimization framework that integrates greedy suspicious-monitoring-point allocation, DRL-based UAV trajectory and task offloading optimization, and convex optimization for UAV computational resource allocation. Unlike previous studies assuming static task assignments, the greedy suspicious-monitoring-point allocation in our framework incorporates dynamic task generation and load-balancing awareness, enabling the system to adapt to real-time monitoring requirements while distributing tasks more evenly among USVs. The combination of greedy, DRL, and convex optimization simultaneously addresses energy efficiency, coverage, and load balancing, which cannot be achieved by prior hybrid approaches combining only two techniques, demonstrating the framework's uniqueness and practical applicability in intelligent aquatic monitoring scenarios.

3. System model

As shown in Fig. 1, we consider an intelligent lake monitoring system consisting of M USVs, a reconnaissance UAV, and a UAV equipped with a MEC module. The set of USVs is denoted as $\mathcal{M} = \{1, \ldots, M\}$. A Time Division Multiple Access (TDMA) mechanism is adopted in this paper. The continuous time is divided into equal-length time slots, denoted as $\mathcal{T} = 1, \ldots, T$. Communication scheduling is performed at the granularity of time slots. Within each slot, multiple USVs may concurrently access the UAV, and the UAV multiplexes the total bandwidth among them. The positions of the UAV and USVs are assumed to be unchanged within each time slot, and they are updated at the end of each time slot according to the flight and navigation decisions. Before the system begins to operate, the reconnaissance UAV is first deployed to scan the entire lake surface to identify and dynamically generate a set of suspicious task points. Meanwhile, the UAV equipped with the MEC module provides real-time computational and communication support to the USVs, with its position dynamically adjusted based on the distribution and communication status of the USVs to maintain efficient coverage and computational performance. For example, a continuous anomalous bubble plume on the lake surface often indicates potential underwater disturbances, such as gas seepage, pipeline leakage, or abnormal biological activity. Therefore when the reconnaissance UAV detects a continuous anomalous bubble plume at a specific position on the lake surface, it designates that position as a suspicious task point. Therefore, a suspicious task point is essentially a location where potential anomalous events could occur. These suspicious task points are then assigned to different USVs, and the USVs will visit them sequentially according to the designated strategy for further verification. When a USV arrives at a suspicious task point, it performs close-range high-precision detection at that location to determine whether a genuine anomaly exists. For instance, the USV may measure turbidity, conductivity, or harmful substance concentration at the suspicious point to assess whether a pollution event is indeed present. If the detection confirms that an actual anomaly exists, a task is immediately generated at that position, triggering subsequent processing procedures.

After the initial suspicious task points and their assignments have been determined, each USV employs the classical Ant Colony Optimization (ACO) algorithm to plan the visiting sequence of its assigned suspicious points. Each USV strictly follows this planned sequence to sequentially navigate to the suspicious task points and, upon arrival, trigger and complete the corresponding monitoring tasks. Within each time slot, task sensing and generation, UAV trajectory planning, task offloading decision-making, and computation execution are synchronously performed. We assume that the data size of each task is sufficiently small, such that the generated tasks can complete both computation and communication operations within the same time slot, thereby ensuring the temporal independence and modeling feasibility of the system. Therefore, each USV is assumed to handle only a single task within each time slot. If tasks extend across multiple time

slots, they may introduce task backlog, queue accumulation, and inter-slot coupling of communication and computation resources, requiring offloading decisions to consider long-term delay and system stability. Furthermore, during task execution, each USV continues moving along its planned trajectory, and task computation and communication are performed concurrently with navigation. Under this assumption, the impact of mobility on communication and computation can be neglected within a single time slot. This constitutes a different class of optimization problem.

During the cruising process of the USVs, new tasks may be dynamically generated. On the one hand, the reconnaissance UAV may detect new suspicious task points, while continuously patrolling and scanning the lake area. Then, the system assesses the additional travel cost for each USV to reach suspicious task points, and assigns the most suitable USV to visit then by considering the current load of each USV. On the other hand, USVs can detect anomalies via their onboard sensing modules and autonomously generate new tasks during navigation.

In our energy modeling, we explicitly consider the energy consumption of both USVs and UAVs and divide it into different components. Specifically, the total energy consumption of each USV includes propulsion energy, communication energy, and computation energy, while for the UAV, its energy model primarily includes the energy consumed for task computation. This distinction allows each type of energy consumption to be clearly represented in its respective optimization objectives, ensuring a comprehensive evaluation of the overall system energy consumption.

It should be noted that, for modeling tractability and to focus on the core joint optimization problem, the USV energy model assumes constant-speed hydrodynamic drag, and the channel model adopts a deterministic Line-of-Sight (LoS) path loss. Complex effects such as acceleration/deceleration dynamics, multipath fading, and water surface reflections are not explicitly modeled. These simplifications help maintain analytical feasibility while not affecting the validity of the proposed optimization framework in theoretical analysis and simulation results.

Denote the sequence of suspicious task points allocated to USV m at time slot t by $S_m(t)$. The position of USV m at time slot t is given by $l_m^{usv}(t) = \left(x_m(t),\, y_m(t),\, 0\right)$. We assume that one USV can generate at most one task at its current position with probability $\rho_m(t)$. Therefore, the task generated by USV m during time slot t can be expressed as:

$$W_m(t) = \left\{ (D_m(t), C_m(t)) | X_m(t) = 1 \right\}, \tag{1}$$

where $X_m(t) \sim \text{Bernoulli}\left(\rho_m(t)\right)$, and

$$\rho_m(t) = \begin{cases} \rho_1, & \text{if } l_m^{usv}(t) \text{ is a normal position.} \\ \rho_2, & \text{if } l_m^{usv}(t) \text{ is a suspicious task point.} \end{cases} \tag{2}$$

where $X_m(t) = 1$ indicates that a task is generated by USV m at time slot t, and $X_m(t) = 0$ indicates no task generation.

When the task $W_m(t)$ needs to be processed, we adopt a partial task offloading mechanism in this paper. Specifically, the USV m can choose to execute $W_m(t)$ locally or offload part of task to the UAV for execution, when USV m is within the serving area of the UAV. The partial offloading mechanism enables the acceleration of task processing and energy optimization at the local side.

3.1. USV motion model

We assume that each USV m moves at a constant speed V_m between different suspicious task points. According to its task points sequence $S_m(t)$, the next target suspicious task point to which it will move in the current time slot is given by:

$$l_m^{\text{next}}(t) = \left(x_m^{\text{next}}(t),\, y_m^{\text{next}}(t),\, 0\right) \in S_m(t), \tag{3}$$

where $l_m^{\text{next}}(t)$ represents the spatial coordinate of the next suspicious task point to be visited by USV m. The position of the next suspicious task point is updated as the USV reaches a new suspicious task point. Therefore, at time slot $t+1$ the position model of the USV m is updated as follows:

$$\begin{cases} x_m(t+1) = x_m(t) + \tau v_m \cos\theta_m(t), \\ y_m(t+1) = y_m(t) + \tau v_m \sin\theta_m(t). \end{cases} \tag{4}$$

where $\theta_m(t) = \text{atan2}\left(y_m^{\text{next}}(t) - y_m(t),\ x_m^{\text{next}}(t) - x_m(t)\right)$ represents the heading direction of the USV m, and τ is the duration of a time slot.

Generally, the hydrodynamic drag encountered by a USV during navigation is much greater than the aerodynamic drag. Therefore, in this paper we only consider the hydrodynamic drag. According to the energy consumption model, for USV m, the hydrodynamic drag can be expressed as:

$$R[V_m] = \frac{1}{2}\varrho_w C_f \Omega_m v_m^2, \tag{5}$$

where ϱ_w is the water density, C_f is the friction coefficient, and Ω_m is the wetted surface area of USV m. Consequently, the propulsion energy consumption of USV m per unit distance can be expressed as:

$$E_m^{\text{unit}} = \frac{R[v_m]}{\eta_{\text{prop}}}, \tag{6}$$

where η_{prop} denotes the propulsion efficiency of the USV.

3.2. UAV mobility model

Assume that at the current time slot t, the position of UAV is $l_u^{\text{uav}}(t) = \left(x_u(t),\, y_u(t),\, z_u(t)\right)$, with horizontal yaw angle $\theta_u^h(t) \in [0, 2\pi)$, pitch angle $\theta_u^p(t) \in [-\pi/2, \pi/2]$, and flight speed $v_u(t) \in [0, v_{\max}]$. Then, the position of UAV at time slot $t+1$ can be expressed as:

$$\begin{cases} x_u(t+1) = x_u(t) + \tau v_u(t) \cos\theta_u^p(t) \cos\theta_u^h(t), \\ y_u(t+1) = y_u(t) + \tau v_u(t) \cos\theta_u^p(t) \sin\theta_u^h(t), \\ z_u(t+1) = z_u(t) + \tau v_u(t) \sin\theta_u^p(t). \end{cases} \tag{7}$$

Assume that the maximum communication pitch angle of UAV is $\theta_u^c(t) \in \left(0, \frac{\pi}{2}\right)$. Then, at time slot t, the maximum communication distance is given by: $D_{\max}(t) = \frac{z_u(t)}{\cos\theta_u^c(t)}$. For USVs located within the UAV's communication range, the UAV can connect with them and process the portion of tasks offloaded by these USVs. Let $\delta_{u,m}(t) \in \{0, 1\}$ indicate whether UAV can provide service to USV m during the current time slot t. Specifically, $\delta_{u,m}(t) = 1$ means that USV m is within the communication range of the UAV at time slot t, whereas $\delta_{u,m}(t) = 0$ indicates that it is out of the UAV's communication range at time slot t.

3.3. USV local computation model

For task processing, we adopt a partial task offloading strategy:

$$\varphi(t) = \{\varphi_{m,u}(t) \mid 1 \le m \le M\}, \tag{8}$$

where $\varphi_{m,u}(t) \in [0, 1]$ denote the proportion of the task offloaded from USV m to UAV at time slot t. Specifically, $\varphi_{m,u}(t) = 0$ indicates that the task is entirely processed locally on the USV, while $\varphi_{m,u}(t) = 1$ means the task is fully offloaded to the UAV for computation. Therefore, at time slot t, the delay incurred by USV m to process its task can be expressed as:

$$T_m^{\text{local}}(t) = \frac{(1 - \varphi_{m,u}(t)) D_m(t) C_m(t)}{f_m^{\text{usv}}}, \tag{9}$$

where f_m^{usv} denotes the computational capability of USV m. Thus, the local computation energy of USV m can be expressed as:

$$E_m^{\text{local}}(t) = k_{\text{usv}} \left(f_m^{\text{usv}}\right)^3 T_m^{\text{local}}(t), \tag{10}$$

where k_{usv} denotes the computation energy coefficient of USV.

3.4. Edge offloading model

Since there are very few obstacles over urban lakes, the link between the UAV and the USV can be considered a standard LoS link. Denote the distance between USV m and the UAV at time slot t by

$$D_{m,u}(t) = \sqrt{(x_u(t) - x_m(t))^2 + (y_u(t) - y_m(t))^2 + z_u(t)^2}, \tag{11}$$

and if $D_{m,u}(t) \le D_{\max}(t)$, then $\delta_{u,m}(t) = 1$.

The channel gain between USV m and the UAV can be expressed as:

$$g_{m,u}(t) = g_0 \, D_{m,u}(t)^{\varpi}, \tag{12}$$

where g_0 denotes the channel power gain at a reference distance of one meter. ϖ denotes the path-loss exponent, which can be adjusted to model different channel attenuation characteristics.

Assuming that the bandwidth of the UAV is B_0, during time slot t, the UAV's bandwidth is equally shared among the USVs connected to it at that time. Therefore, the bandwidth allocated to USV m is:

$$B_m(t) = \frac{B_0}{\sum_{m=1}^{M} \delta_{u,m}(t)}. \tag{13}$$

Therefore, the data transmission rate between USV m and the UAV is given by:

$$R_{m,u}(t) = B_m(t) \log\left(1 + \frac{p_m g_{m,u}(t)}{\sigma^2}\right), \tag{14}$$

where p_m denotes the transmit power of USV m, and σ^2 denotes the power of additive white Gaussian noise.

Assume that at time slot t, the amount of task from USV m to be offloaded is denoted as $\varphi_{m,u}(t) D_m(t)$. Therefore, the transmission delay required for uploading its task is:

$$T^{\text{tran}}_{m,u}(t) = \frac{\varphi_{m,u}(t) \, D_m(t)}{R_{m,u}(t)}, \tag{15}$$

and the corresponding energy consumption for uploading the task is:

$$E^{\text{tran}}_{m,u}(t) = p_m \, T^{\text{tran}}_{m,u}(t). \tag{16}$$

The UAV can process tasks offloaded from multiple USVs at the same time. The computation resource allocation of the UAV at time slot t can be expressed as:

$$f(t) = \{ f_{u,m}(t) \mid 1 \le m \le M \}, \tag{17}$$

where $f_{u,m}(t)$ denotes the computation resource allocated by UAV to the task uploaded by USV m at time slot t. Accordingly, the computation delay for UAV processing USV m's task is:

$$T^{\text{uav}}_{m}(t) = \frac{\varphi_{m,u}(t) \, D_m(t) \, C_m(t)}{f_{u,m}(t)}, \tag{18}$$

where $C_m(t)$ denotes the required CPU cycles per bit for USV m's task.

The energy consumption of the UAV processing USV m's task is:

$$E^{\text{uav}}_{m}(t) = k_{\text{uav}} \left(f_{u,m}(t)\right)^3 T^{\text{uav}}_{m}(t), \tag{19}$$

where k_{uav} denotes the computation energy coefficient of the UAV. Since the data size of execution result returned to USV m is negligible, it is ignored in this work. Therefore, the processing delay for the task offloaded by USV m to the UAV during time slot t is:

$$T^{\text{mec}}_{m}(t) = T^{\text{tran}}_{m,u}(t) + T^{\text{uav}}_{m}(t), \tag{20}$$

and the corresponding energy consumption is

$$E^{\text{mec}}_{m}(t) = E^{\text{tran}}_{m,u}(t) + E^{\text{uav}}_{m}(t). \tag{21}$$

3.5. Load-balancing USV assignment for new task points

As mentioned earlier, the reconnaissance UAV may detect new suspicious task points at different time slots. In this case, the system needs to determine which USVs are assigned to detect the new task points. If the task assignment only focuses on minimizing the additional energy needed to reach the task point, certain USVs may be scheduled more frequently, leading to unbalanced workload and energy consumption among USVs. This imbalance can greatly degrade the overall efficiency of the system in terms of operability and robustness. To address this issue, we put forward a load-balancing USV assignment strategy for newly detected suspicious task points, aiming to strike a balance between the additional energy consumption and workloads among different USVs.

Note that the energy consumed during the navigation of a USV is much higher than that required for task computation. Therefore, we only take the navigation energy into account when a USV moves for the new suspicious task point. Assume that the UAV detects a new suspicious task point $l_{\text{new}}(t)$ at time slot t that is assigned to the USV m, the additional energy consumption for m can be given as:

$$\Delta E_m(t) = E^{\text{unit}}_{m} \, \Delta \text{L}_m(t), \tag{22}$$

where $\Delta \text{L}_m(t)$ denotes the extra path cost for USV m to reach the new suspicious task point. The additional energy consumption $\Delta E_m(t)$ should satisfy the following constraint:

$$E^{\text{remain}}_{m}(t) - \Delta E_m(t) \ge E_{\epsilon}, \tag{23}$$

where $E^{\text{remain}}_{m}(t)$ represents the remaining energy of USV m at the current time slot. The value of $E^{\text{remain}}_{m}(t)$ can be initialized according to each USV's initial path planning and updated dynamically during operation. E_{ϵ} is the energy consumption threshold of the USV. A penalty term $\psi_m(t)$ is defined to characterize the load balance of USV m [26]:

$$\psi_m(t) = \frac{\left(\sum_{m=1}^{M} N_m(t)\right)^2}{M \sum_{m=1}^{M} \left(N_m(t)\right)^2}, \tag{24}$$

where $N_m(t)$ represents the number of suspicious task points assigned to USV m at time slot t after tentatively allocating the current suspicious monitoring point to it. The load-balancing coefficient $\psi(t) \in [\frac{1}{M}, 1]$ measures the overall load balance among all USVs at time t. To evaluate candidate assignments of suspicious monitoring points, we define $\psi_m(t)$ as the resulting global fairness index *if* the task were assigned to USV m. Therefore, while $\psi(t)$ reflects the actual system state, a higher $\psi_m(t)$ indicates a potentially better assignment strategy and the overall cost function can be expressed as:

$$F_m(t) = \frac{\Delta E_m(t)}{\psi_m(t)}. \tag{25}$$

4. Problem formulation

The optimization problem in this paper includes two distinct parts. The first part addresses the USV assignment for multiple suspicious task points dynamically detected during system operation. The goal of this USV assignment problem is to minimize the required energy consumption while maintaining load-balancing among all the USVs.

$$J_1 : \; m^*(t) = \arg\min_{m \in \mathcal{M}} F_m(t), \qquad \forall t \in \mathcal{T} \tag{26}$$

$$\text{s.t.} \quad E^{\text{remain}}_{m}(t) - \Delta E_m(t) \ge E_{\epsilon}, \quad \forall m \in \mathcal{M} \tag{26a}$$

where $F_m(t)$ is the cost function defined in (25).

The second part aims to jointly optimize the UAV flight trajectory and task offloading strategy so as to maximize the overall coverage

of USVs by UAV during the system operation period, while simultaneously minimizing the task response latency and computational energy consumption.

For each task generated by a USV m, the required computation delay can be expressed as:

$$T_m(t) = \max\left(T_m^{\text{local}}(t), T_m^{\text{mec}}(t)\right), \tag{27}$$

and the energy consumption is:

$$E_m(t) = E_m^{\text{local}}(t) + E_m^{\text{mec}}(t). \tag{28}$$

The overall optimization objective can be expressed as:

$$\begin{aligned} J_2: \max_{\substack{\varphi(t),\, f(t),\\ v_u(t),\, \theta_u^p(t),\, \theta_u^h(t)}} & \sum_{t=1}^{T} \eta(t) = \zeta_1 \left(\sum_{t=1}^{T} \frac{\sum_{m=1}^{M} \delta_{u,m}(t)}{M} \right) \\ & - \zeta_2 \sum_{t=1}^{T} \sum_{m=1}^{M} \left(\alpha \frac{T_m(t)}{T_{\max}} + (1-\alpha) \frac{E_m(t)}{E_{\max}} \right) \end{aligned} \tag{29}$$

$$\text{s.t.} \quad \sum_{m=1}^{M} f_{u,m}(t) \le f_{\text{UAV}}, \tag{29a}$$

$$\varphi_{m,u}(t) \in [0, 1], \tag{29b}$$

$$0 \le v_u(t) \le v_{\max}, \tag{29c}$$

$$\theta_u^h(t) \in [0,\, 2\pi), \tag{29d}$$

$$\theta_u^p(t) \in \left[-\frac{\pi}{2},\, \frac{\pi}{2}\right], \tag{29e}$$

$$0 \le x_u(t) \le x_{\max}, \tag{29f}$$

$$0 \le y_u(t) \le y_{\max}, \tag{29g}$$

$$0 \le z_u(t) \le z_{\max}, \tag{29h}$$

where $\varphi(t)$ denotes the offloading decision of the USV, $f(t)$ represents the computation resource allocation of the UAV, $v_u(t)$ denotes the UAV speed, and $\theta_u^h(t)$ and $\theta_u^p(t)$ represent the horizontal yaw angle and vertical pitch angle. $\delta_{u,m}(t)$ indicates whether USV m is within the UAV's coverage at time t ($\delta_{u,m}(t) = 1$ if covered, 0 otherwise). The coverage term $\frac{\sum_{m=1}^{M} \delta_{u,m}(t)}{M}$ thus represents the proportion of USVs currently covered by the UAV, ranging from 0 (no USVs covered) to 1 (all USVs covered). ζ_1 and ζ_2 are weighting factors that balance the trade-off between the coverage benefit and the delay–energy cost, α denotes the weight coefficient between delay and energy consumption, $T_{\max}$ and $E_{\max}$ are the reference maximum delay and maximum energy for task execution in the system, i.e., the maximum time and energy a single task could consume under worst-case conditions. To unify the scales of delay and energy for weighting, the delay metric is normalized by dividing by $T_{\max}$, and the energy metric is normalized by dividing by $E_{\max}$ before applying the weight α. $V_{\max}, x_{\max}, y_{\max}, z_{\max}$ are the corresponding maximum limits, and f_{UAV} is the total computation resource of the UAV.

5. Algorithm implementation

For the optimization problem J_1, an algorithm named GDSTA and presented in Algorithm 1, is designed to assign dynamically generated suspicious task points to the most suitable USV, aiming to minimize the total system energy consumption while maintaining the load-balancing of task allocation among all USVs. For optimization problems J_2, which jointly involve UAV trajectory planning, task offloading, and computational resource allocation, the problem exhibits strong coupling among multiple decision variables and is therefore highly complex. To address this challenge, a Joint Aerial Trajectory and Offloading algorithm based on Deep Reinforcement Learning (JATO-DRL) is proposed to jointly optimize the UAV flight trajectory and task offloading decisions. Owing to the convexity of the resource allocation subproblem, a convex optimization method is employed to obtain the optimal UAV computational resource allocation after the offloading decision is determined at each iteration.

5.1. GDSTA algorithm description

When a new suspicious task point $l_{\text{new}}(t)$ is detected by the reconnaissance UAV at time slot t, the GDSTA algorithm dynamically evaluates the impact of inserting this task into each USV's current route. For USV m, the additional travel distance $\Delta \text{D}_m(t)$ is determined by traversing all possible insertion positions within the unvisited portion of its route and selecting the one that yields the minimal incremental path length:

$$\Delta \text{L}_m(t) = \min_i \Big(D(P_i, l_{\text{new}}(t)) + D(l_{\text{new}}(t), P_{i+1}) - D(P_i, P_{i+1}) \Big), \tag{30}$$

where P_i and P_{i+1} denote two consecutive task points in USV m's route, and $D(\cdot, \cdot)$ represents the Euclidean distance.

From a computational perspective, evaluating a single insertion for USV m requires $O(1)$ distance computations, and there are at most $|\mathcal{R}_m(t)|$ candidate insertion positions within the remaining route. Therefore, the worst-case time complexity for computing $\Delta \text{D}_m(t)$ is $O(|\mathcal{R}_m(t)|)$. Since this operation is performed for all M USVs, handling a newly detected task point incurs a total worst-case complexity of:

$$O\Big(\sum_{m=1}^{M} |\mathcal{R}_m(t)| \Big), \tag{31}$$

which simplifies to $O(MN)$ when each USV maintains $O(N)$ unvisited task points.

Compared with large-scale combinatorial optimization methods that require global re-planning of all tasks, the proposed insertion mechanism exhibits linear complexity with respect to both the number of USVs and task points, leading to lightweight computational overhead and fast online updates under dynamically arriving tasks. Therefore, GDSTA offers clear advantages in lake surface inspection scenarios where both real-time capability and large-scale deployment are required.

Algorithm 1 summarizes the procedure of dynamic task insertion and allocation (i.e., GDSTA algorithm).

Algorithm 1: Greedy Dynamic Task Allocation (GDSTA) Algorithm

Input: $S(t)$: task sequences of all USVs;
$l_{\text{new}}(t)$: position of new suspicious task point;
$E_{\text{remain}}(t)$: remaining energy of USVs;
$\psi(t)$: load factor of USVs
Output: Updated $S(t)$, selected USV $m^*(t)$, insert position $i^*(t)$, minimum cost $F_{\min}(t)$

$F_{min}(t) \leftarrow \infty$;
$m^*(t) \leftarrow -1$;
$i^*(t) \leftarrow -1$;
for each USV m **do**
  Obtain current position $l_m^{\text{usv}}(t)$;
  for each insertion point i in unvisited tasks **do**
    Compute incremental distance $\Delta L_m(t)$ using Eq. (30);
    Compute extra cost $F_m(t)$ using Eq. (25);
    if $F_m(t) < F_{min}$ *and* (26a) **then**
      Update $F_{\min}(t) \leftarrow F_m(t)$, $m^* \leftarrow m$, $i^* \leftarrow i$;
Insert $S_{\text{new}}(t)$ into $S_m(t)$ at position $i^*(t)$;
return $S(t)$, $m^*(t)$, $i^*(t)$;

5.2. JATO-DRL algorithm

For UAV trajectory optimization and task offloading decision making, traditional optimization algorithms often struggle to simultaneously minimize task execution delay and energy consumption while maximizing UAV coverage in dynamic environments. In particular, under complex conditions such as time-varying channels, fluctuating task loads, and limited computational resources, these methods usually

fail to achieve global optimality. To address this issue, we propose the JATO-DRL algorithm, which enables intelligent decision-making and cooperative optimization for UAVs. The overall optimization objective is formulated as a Markov Decision Process (MDP), whose key components include the state space S_t, action space A_t, and reward function R_t.

State Space S_t: The environment state contains all the information required for DRL, which is jointly composed of the states of the UAV and USVs, and dynamically changes with each time slot t. The state space can be expressed as: $S_t = \{s_m^{usv}(t),\ s_u^{uav}(t) \mid m \in \{1, \dots, M\}\}$.

The state of USV m is defined as $s_m^{usv}(t) = \{l_m^{usv}(t),\ f_m^{usv},\ V_m,\ D_m(t),\ C_m(t),\ \delta_{u,m}(t),\ j_m^{usv}(t)\}$, where each element represents the current position of the USV, its available computational resources, velocity, task data size, computational cycles required for the task, whether it is within the UAV's coverage area, and the target suspicious monitoring point location, respectively.

The UAV state is defined as: $s_u^{uav}(t) = \{l_u^{uav}(t),\ f_{uav}\}$ where each element denotes the current position of the UAV and its available computational resources, respectively.

Action Space A_t: In this study, the action space of the UAV agent includes its flight trajectory control variables $v_u(t)$, $\theta_u^p(t)$, $\theta_u^h(t)$, and the task offloading decision $\varphi(t)$ for each USV. Therefore, the action space is defined as: $A_t = \{v_u(t),\ \theta_u^p(t),\ \theta_u^h(t),\ \varphi(t)\}$.

Reward Function R_t: When the agent takes an action in a given state, it receives an immediate reward. To minimize the weighted sum of task execution delay and energy consumption while maximizing the UAV coverage rate, the reward function is defined as: $R_t = \beta_1 \eta(t) - \beta_2 \eta_{\text{penalty}}(t)$, where β_1 and β_2 represent the weighting coefficients for the coverage benefit and the combined delay-energy-constraint penalty, respectively.

Constraint Handling: During the learning process, constraints (29a)–(29h) are enforced through a hybrid mechanism. Constraint (29a) defines a convex feasible region for UAV computation resource allocation and is always satisfied when generating allocation actions. Constraints (29b)–(29e) are interval constraints related to UAV motion and orientation, and continuous actions are projected (clipped) into their respective feasible ranges at each step. Spatial boundary constraints (29f)–(29h) are treated as hard limits: the UAV state is clipped back into the valid region whenever a violation occurs, and such violations contribute to the penalty term $\eta_{\text{penalty}}(t)$ in the reward. This mechanism ensures physical feasibility while preserving the stability of policy learning.

In this paper, we propose the JATO-DRL algorithm, which is built upon the Soft Actor-Critic (SAC) framework. The algorithm is capable of performing joint optimization of policy learning and value estimation in continuous action spaces. Its core modules include a policy network (Actor) and a double Q-value network (Critic). The policy network takes the current environment state S_t as input, extracts high-level features through two fully connected layers, and outputs the mean and logarithm of standard deviation of a Gaussian policy:

$$\pi_\theta(A_t \mid S_t) = \mathcal{N}\left(\mu_\theta(S_t), \sigma_\theta^2(S_t)\right). \tag{32}$$

To ensure differentiability, the reparameterization trick is employed to sample actions:

$$A_t = \tanh\Big(\mu_\theta(S_t) + \sigma_\theta(S_t) \odot \epsilon\Big), \quad \epsilon \sim \mathcal{N}(0, I). \tag{33}$$

The value networks adopt a double Q structure to mitigate overestimation bias. Two Q-networks estimate the state–action value separately: $Q_i(S_t, A_t), i \in \{1, 2\}$. To further improve training stability, each Q-network is implemented in two forms: (1) an online Q-network $Q_i(S_t; \theta_i)$, whose parameters are updated directly via gradient descent, and (2) a target Q-network $Q_i(S_t; \theta_i')$, which is a slowly updated copy of the online network. The target Q-networks are used exclusively for computing the temporal-difference (TD) targets, thereby reducing oscillation during training. Using the target networks, the temporal-difference target for the next state is defined as:

$$y_t = R_t + \delta\Big[\min_{i=1,2} Q_i'(S_{t+1}, A_{t+1}) - \lambda \log \pi_\theta(A_{t+1}|S_{t+1})\Big], \tag{34}$$

where λ is the temperature parameter that balances exploration and exploitation.

The critic loss function is defined as:

$$L_{\text{critic}} = \mathbb{E}_{(S_t, A_t)\sim D}\left[\sum_{i=1}^{2}\big(Q_i(S_t, A_t) - y_t\big)^2\right]. \tag{35}$$

The actor is optimized to maximize the expected Q-value while penalizing low-entropy actions:

$$L_{\text{actor}} = \mathbb{E}_{S_t\sim D}\left[\lambda \log \pi_\theta(A_t|S_t) - \min_{i=1,2} Q_i(S_t, A_t)\right]. \tag{36}$$

The temperature parameter λ is automatically adjusted to maintain a target entropy level H_{target}:

$$L_\lambda = \mathbb{E}_{A_t\sim\pi_\theta}\Big[-\lambda(\log \pi_\theta(A_t|S_t) + H_{\text{target}})\Big]. \tag{37}$$

The target Q-networks are updated using a soft update strategy:

$$\theta_i' \longleftarrow \xi\theta_i + (1-\xi)\theta_i', \quad i = 1, 2, \tag{38}$$

where $\xi \in (0, 1)$ is the update coefficient, θ_i denotes the parameters of the online Q-network, which is directly updated by gradient descent. In contrast, θ_i' represents the parameters of the target Q-network, which are updated slowly using the soft update rule in (38) to provide a stable target for temporal-difference learning.

Algorithm 2: Joint Aerial Trajectory and Offloading based on Deep Reinforcement Learning (JATO-DRL) Algorithm

Input: N_{episode}: maximum training episodes;
N_{step}: maximum steps per episode;
γ: discount factor;
τ: soft update coefficient;
D: replay buffer;
environment **env**.
Output: Trained actor network π_θ and critic networks Q_{ϕ_1}, Q_{ϕ_2}.

Initialize actor π_θ, critic Q_{ϕ_1}, Q_{ϕ_2}, target critic $\bar{Q}_{\phi'} \leftarrow Q_\phi$;
Initialize replay buffer $\mathcal{D}$;
for *episode* $= 1$ **to** $N_{episode}$ **do**
  Reset environment **env** and get initial state S_0;
  Initialize cumulative reward $R_{\text{ep}} \leftarrow 0$;
  for $t = 1$ **to** N_{step} **do**
    Select action A_t using Eq. (33);
    Execute action A_t in **env**, observe $(S_{t+1}, R_t, done_t)$;
    Update the suspicious task queue $S(t)$ for all USV using Algorithm 1;
    Store $(S_t, A_t, S_{t+1}, R_t, done_t)$ into replay buffer $\mathcal{D}$;
    if $|D| > N_{min}$ **then**
      Sample a mini-batch $(S, A, S', R, done)$ from $\mathcal{D}$;
      Compute target value using Eq. (34);
      Update critics by minimizing Eq. (35);
      Update actor by minimizing Eq. (36);
      Adjust temperature parameter α via Eq. (37);
      Soft-update target critics using Eq. (38);
    $R_{\text{ep}} \leftarrow R_{\text{ep}} + R_t$;
    if $done_t$ **then**
      break;
  Save model parameters if $R_{\text{ep}} > R_{\text{best}}$;
  Record loss curves and α evolution for visualization;
return π_θ^*, $Q_{\phi_1}^*$, $Q_{\phi_2}^*$

5.3. UAV computing resource allocation under fixed offloading decisions

After the task offloading decisions have been determined using the JATO-DRL algorithm and the tasks have been uploaded from the USVs to the UAV, the next step is to allocate the limited computational resources of UAV among its assigned tasks efficiently. Let $U(t)$ denote the set of USVs that uploaded tasks to the UAV at time slot t, and define

$$f(t) = \{ f_{u,m}(t) \mid m \in U(t) \}, \tag{39}$$

where $f_{u,m}(t) > 0$ represents the computation resource allocated by the UAV to the task offloaded by USV m in slot t.

The slot-level optimization problem for the UAV is formulated as:

$$\min_{f(t)} \; G(f(t)) := \sum_{m \in U(t)} g_m\big(f_{u,m}(t)\big) = \sum_{m \in U(t)} \Big[\alpha_2 \frac{\varphi_{m,u} D_m(t) C_m(t)}{f_{u,m}(t)} + (1-\alpha_2) k_{\text{uav}} \varphi_{m,u} D_m(t) C_m(t) \big(f_{u,m}(t)\big)^2 \Big] \tag{40}$$

$$\text{s.t.} \quad \sum_{m \in U(t)} f_{u,m}(t) \le f_{\text{uav}}, \tag{40a}$$

$$f_{u,m}(t) > 0, \quad \forall m \in U(t). \tag{40b}$$

We next show that problem (40) is strictly convex. For a single task m, define $g_m(f)$ as in (40). It can be verified that $g_m(f)$ is strictly convex for $f > 0$, since its second derivative is strictly positive. As the objective is separable across tasks, the full objective $G(f(t)) = \sum_{m \in U(t)} g_m(f_{u,m}(t))$ remains strictly convex. Meanwhile, constraint (40a) is linear and (40b) defines a convex domain, thus problem (40) is a strictly convex optimization problem and admits a unique global optimum.

To compute the solution, consider the Lagrangian:

$$\mathcal{L}(f(t), \lambda) = \sum_{m \in U(t)} g_m(f_{u,m}(t)) + \lambda \left(\sum_{m \in U(t)} f_{u,m}(t) - f_{\text{uav}} \right), \tag{41}$$

where $\lambda \ge 0$ is the dual variable. The KKT stationarity condition yields a monotonic equation in $f_{u,m}(t)$ whose unique positive root can be written as:

$$f^*_{u,m}(t) = \chi_m(\lambda). \tag{42}$$

The optimal λ^* enforces the total capacity constraint:

$$\sum_{m \in U(t)} \chi_m(\lambda^*) = f_{\text{uav}}. \tag{43}$$

Since $\chi_m(\lambda)$ is strictly decreasing in λ, λ^* can be efficiently obtained using a one-dimensional bisection search.

6. Simulation experiments and result analysis

The experimental environment is based on the Python 3.10 platform with the PyTorch 2.0.1 framework. In the simulation setup, the area size is set to 200×200×100 m^3, with 75 initial suspicious task points, 5 to 9 USVs, 1 UAV, and the task generation probability of each USV ranging from 0.5 to 0.9. The key DRL training settings are as follows: the Actor network has two fully connected layers (256, 128) outputting mean and log standard deviation, and the Critic is a double Q-network with two hidden layers per network (256, 128); the learning rate is 3e-4, soft update coefficient $\tau = 0.005$, batch size 256, replay buffer size 10^6, total training episodes 1000, and maximum steps per episode 500–1000; random seeds for NumPy, PyTorch, and the environment can be fixed to ensure reproducibility; during training, the best-performing model is saved according to the cumulative reward.

Each monitoring task is assumed to correspond to a single data packet of approximately 4–5 Mbits, which is consistent with the typical bitrate range of compressed 720p video streams under H.264/H.265 encoding (about 2.5–6 Mbps). In the simulation, each time slot is set to 1 s. Under this time slot length, for a 5 Mbits task, the worst-case local processing time on a USV is approximately 0.67 s; if offloaded to the UAV, the combined transmission and computation delay is approximately 0.82 s. Thus, each task can be transmitted and processed within a single time slot, ensuring the feasibility of real-time task handling.

Table 1
Simulation parameters.

Parameter description	Value
Task Size	4–5 Mbits
Sensor Task Cycle Frequency	200–400 MHz
Wireless Bandwidth	50–150 MHz
USV Speed	1–3 m/s
USV Computing Capacity	3 GHz
UAV Computing Capacity	15 GHz
Maximum Elevation Angle	60°
USV Upload Power	0.5–0.6 W
UAV flight altitude	30–100 m
Gaussian White Noise PSD	−60 dBm
UAV Speed	4-6 m/s
Channel Power Gain at Reference Distance (1 m)	10^{-4}

Table 2
Algorithm hyperparameters and network overview.

Item	TD3-JATO	A2C-JATO	DDPG-JATO	JATO-DRL
γ	0.9	0.9	0.9	0.9
lr	1e−4	1e−4	1e−4	1e−4
τ	0.005	N/A	0.005	0.005
Policy delay	2	N/A	N/A	N/A
Policy noise	0.2	N/A	0.1	N/A
Noise clip	0.5	N/A	0.5	N/A
Buffer	1e6	N/A	1e6	1e6
Batch	256	N/A	256	256
Actor	Det	Gauss	Det	Gauss
Critic	Twin Q	Value	Single Q	Twin Q
Update	Off-step	On-trajectory	Off-step	Off-step
log_std	N/A	[−20,2]	N/A	[−20,2]
Actor Net	256-128-act	256-128-(μ/σ)	256-128-act	256-128-(μ/σ)
Critic net	256-128-1	256-128-1	256-128-1	256-128-1

In the proposed JATO-DRL algorithm, the Actor network consists of two hidden layers with 256 and 128 neurons, respectively, and outputs both the action mean μ and log standard deviation $\log\sigma$ vectors to construct a Gaussian policy distribution. The Critic network adopts a double Q-network architecture, where each Q-network contains two hidden layers (256 and 128 neurons) and an output layer to estimate the action-value function $Q(s, a)$. The learning rates for the Actor and Critic networks are both set to 0.0001, and the discount factor γ is set to 0.9. Specifics of USVs and UAV are listed in Table 1. The detailed information on algorithm hyperparameters and network structure is provided in Table 2.

The UAV speed range of $4-6$ m/s is selected according to practical operational requirements for supporting USVs in edge computing task offloading. This range ensures reliable communication links between the UAV and multiple USVs and allows timely assistance for task execution, thereby mitigating potential delays in the offloading process. Previous studies have demonstrated that UAVs can assist mobile platforms in edge computing and data collection, emphasizing the importance of selecting appropriate UAV speed and data payload sizes to guarantee real-time system performance [27,28].

6.1. Baseline approaches

For comparative analysis, we consider baseline approaches in three aspects of the system: dynamic suspicious task-point assignment for USVs, task execution with UAV trajectory planning, and a theoretical cloud computing baseline.

First, regarding dynamic suspicious task-point assignment for USVs, we introduce two baseline allocation algorithms:

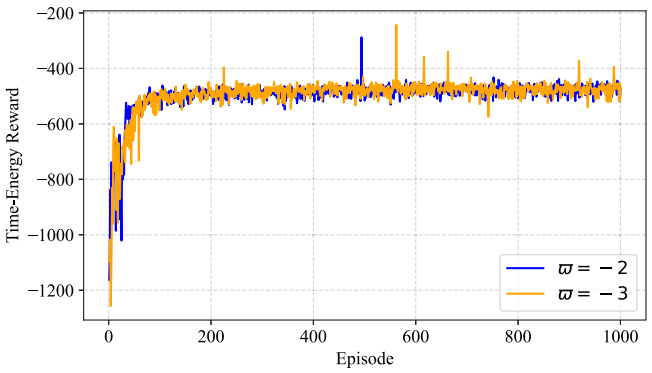


Fig. 2. Reward of JATO-DRL under different path-loss exponents.

- *Random Allocation*: Task points are assigned to USVs in a purely random manner, without considering energy consumption, distance, or task urgency. This serves as a naive baseline for performance comparison.
- *Min-Energy Allocation*: Each suspicious task point is assigned to the USV that requires the least energy to reach and process it, minimizing overall system energy consumption.

Second, for task execution and UAV trajectory planning, we introduce three DRL-based baseline approaches:

- *TD3-Based Joint Offloading and Trajectory Optimization (TD3-JOTO)*: Uses the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm to jointly optimize task offloading decisions and UAV trajectories. The actor network outputs continuous offloading actions, while twin critic networks evaluate action values to mitigate overestimation.
- *A2C-Based Joint Offloading and Trajectory Optimization (A2C-JOTO)*: Utilizes the Advantage Actor-Critic (A2C) framework, where the actor proposes offloading actions and the critic evaluates them using estimated advantage values. Coordinated offloading–trajectory strategies are learned through synchronous updates.
- *DDPG-Based Joint Offloading and Trajectory Optimization (DDPG-JOTO)*: Built on the Deep Deterministic Policy Gradient (DDPG) algorithm, employing a deterministic actor for continuous offloading decisions and a critic network to estimate Q-values. Noise-perturbed actions and experience replay are used to improve performance.

In addition, in many practical scenarios, since the size of the returned results is typically much smaller than that of the uploaded data, the downlink transmission delay T_{downlink} can also be neglected. Therefore, a simplified task latency model can be expressed as:

$$T_{\text{cloud}} = T_{\text{uplink}} + T_{\text{compute}} \tag{44}$$

This simplified model still captures the latency bottleneck of centralized cloud architectures in real-time aquatic monitoring. Considering realistic network bandwidth, task sizes, and cloud processing rates, the resulting latency often exceeds the 1-second time-slot requirement. This analysis demonstrates that centralized cloud architectures may fail to meet stringent timing constraints, thereby further justifying the necessity of edge computing-based solutions.

6.2. Simulation results

Fig. 2 illustrates the convergence behavior of JATO-DRL under different path loss exponents: $\omega = -2$ (corresponding to weaker attenuation, representing open or suburban LoS-dominated environments) and $\omega = -3$ (corresponding to stronger attenuation, closer to urban scenarios with partial blockage or increased effective path loss). The results show that under $\omega = -3$, although the harsher propagation environment increases the variance of the reward signal, leading to noticeable oscillations and spikes during training, the algorithm can still achieve a steeper initial decline within the first few dozen episodes and converge to a long-term Time-Energy reward level comparable to that under $\omega = -2$ (around -500) after approximately 200 episodes. In contrast, the curve for $\omega = -2$ is overall smoother, with smaller amplitude and frequency of fluctuations, demonstrating higher training stability. These observations indicate that the proposed JATO-DRL algorithm exhibits good robustness under different propagation conditions; even under stronger attenuation closer to realistic urban environments, it maintains effective convergence and near-optimal long-term time-energy reward, thereby mitigating the idealized limitations of purely LoS deterministic models. In the simulations, the physical duration of each time slot is set to 1.0 s, corresponding to a reasonable timescale under typical UAV trajectory updates and quasi-static channel assumptions, supporting the algorithm's applicability in quasi-real-time scenarios.

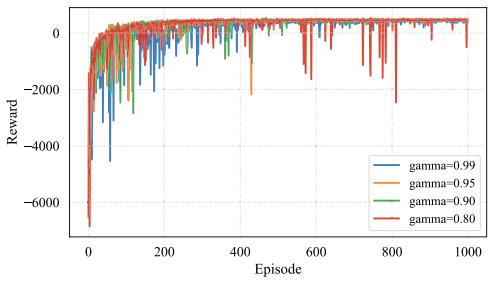


Fig. 3. Training reward curves under different discount factors γ.

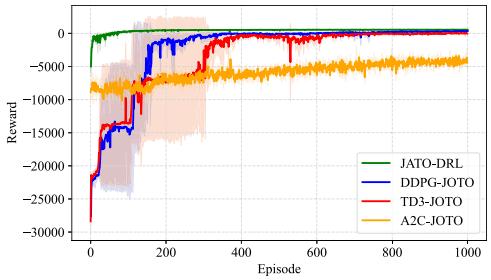


Fig. 4. Learning curves of different algorithms.

Fig. 3 illustrates the variations in training rewards under different discount factors γ and reveals notable differences in convergence behavior across the four configurations. The results show that $\gamma = 0.90$ and $\gamma = 0.95$ achieve the fastest convergence during the early stage of training, with rewards increasing rapidly within the first 100–200 episodes. As training progresses, $\gamma = 0.90$ maintains the most stable performance and achieves the highest long-term reward, whereas $\gamma = 0.95$ and $\gamma = 0.99$ converge more slowly and exhibit larger fluctuations. Overall, $\gamma = 0.90$ provides the best balance between convergence speed and stability.

Fig. 4 illustrates the learning performance of different DRL algorithms during training. The solid curves represent the average reward over three independent runs with different random seeds, while the shaded regions show the corresponding standard deviation. A narrower shaded area indicates greater training stability and lower sensitivity to initialization. In terms of convergence, the proposed JATO-DRL method achieves the highest reward level and converges faster than the baseline algorithms. After approximately 150 episodes, JATO-DRL exhibits both rapid gain and reduced variance, indicating efficient policy learning and strong robustness. In contrast, DDPG-JOTO and

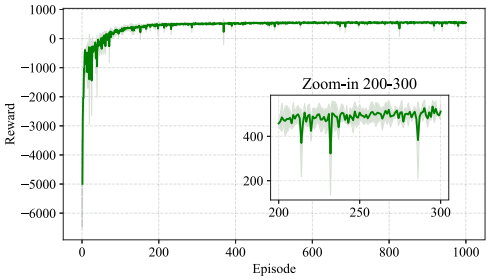


Fig. 5. Training convergence curve of JATO-DRL.

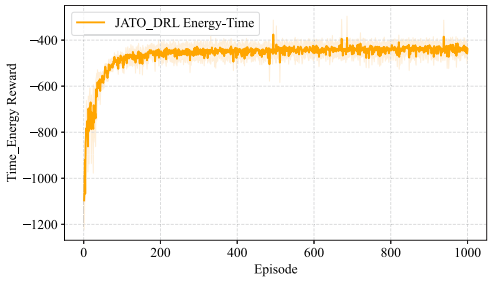


Fig. 6. Convergence curve of the weighted time–energy metric.

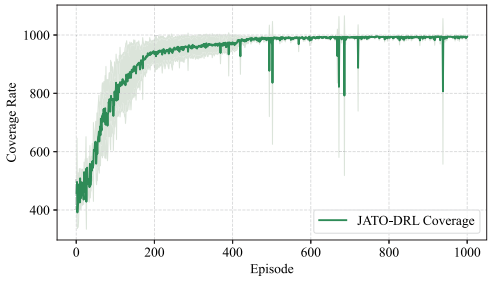


Fig. 7. Convergence curve of the coverage metric.

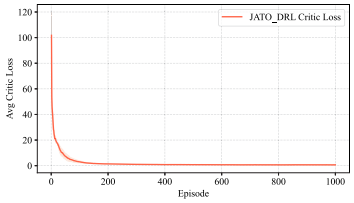


Fig. 8. Convergence of the critic network loss.

TD3-JOTO show slower convergence and larger fluctuations during training, while A2C-JOTO converges to a significantly lower reward level, suggesting weaker learning capability under the same task conditions. These results collectively demonstrate that the proposed method not only converges more efficiently but also maintains superior stability and consistency across random seeds.

Fig. 5 illustrates the training reward curve of the proposed SAC algorithm over 1000 episodes, averaged over three independent runs with different random seeds. In the early stage of training, the reward increases rapidly, indicating that the agent can quickly learn an effective policy. After approximately 200–300 episodes, the reward becomes stable with only minor fluctuations, demonstrating good training stability. The inset (episodes 200–300) highlights the detailed reward variations during the convergence phase. Due to the dynamic nature of the environment, the reward is not completely flat but exhibits certain fluctuations. Moreover, the reward is defined based on weighted coverage, latency, and energy consumption, so higher values correspond to overall improvements across these metrics.

Figs. 6 and 7 jointly illustrate the convergence behavior of the proposed algorithm across multiple performance metrics. As shown in Fig. 6, the weighted time–energy reward increase rapidly during the early training phase and stabilizes after approximately 300 training episodes, indicating that the agent quickly learns an efficient resource allocation policy. The smoothed curve further confirms that the overall convergence process is stable, with only minor fluctuations. Fig. 7 presents the convergence trend of the coverage metric. The coverage steadily increases throughout training and eventually stabilizes at a high level, demonstrating that the agent continuously improves the system's sensing coverage while maintaining favorable delay and energy performance. Taken together, Figs. 6 and 7 show that the proposed algorithm achieves stable and consistent optimization between time-energy efficiency and coverage.

Fig. 8 illustrates the convergence behavior of the critic network during training. The critic loss drops sharply at the beginning, indicating that the value function rapidly adapts to the environment dynamics. After approximately 150 episodes, the loss gradually stabilizes near zero, showing that the critic has effectively learned to approximate the true value function. The smooth and monotonic decline without oscillation suggests that the training process is numerically stable and free from issues such as value divergence or overestimation.

Fig. 9 illustrates the trade-off between latency and energy consumption as the weight factor varies in the objective function. A larger weight emphasizes latency reduction, resulting in more offloading to UAV but higher energy consumption, whereas a smaller weight prioritizes local processing and energy efficiency. Our experiments indicate that once the weight factor becomes too large, the system tends to excessively offload tasks, causing a rapid increase in energy consumption. In this work, we set the weight factor to 0.7 to ensure a more balanced and stable trade-off between latency and energy consumption.

Fig. 10 illustrates the Time-Energy Reward of different algorithms under varying task generation probabilities in a multi-USV–UAV collaborative edge computing scenario. Overall, the system reward of all algorithms decreases with higher task loads due to intensified competition for computing and communication resources. JATO-DRL consistently achieves the highest reward; for example, at a task generation probability of 0.7, its reward is approximately −570, compared with −575, −886, and −1147 for DDPG-JOTO, TD3-JOTO, and A2C-JOTO, respectively. Moreover, the reward of JATO-DRL and TD3-JOTO decreases more smoothly, indicating strong robustness and adaptability, whereas A2C-JOTO shows a sharp decreases under high load and DDPG-JOTO performs moderately. These results demonstrate that JATO-DRL effectively reduces latency and energy consumption while maintaining excellent robustness, making it a reliable task offloading solution for multi-USV–UAV systems.

Fig. 11 shows the Time-Energy Reward of different algorithms under varying bandwidth conditions in the multi-USV–UAV collaborative edge computing scenario. Overall, increasing the available bandwidth increase the Time-Energy reward for all algorithms, as higher bandwidth improves data transmission rates, thereby lowering task offloading delay and energy consumption. JATO-DRL consistently achieves the highest and most stable Time-Energy reward across all bandwidth levels (around −510), demonstrating its robustness and efficient task scheduling under both limited and ample communication resources. In contrast, A2C-JOTO exhibits large fluctuations, particularly under constrained bandwidth, indicating poor adaptability. TD3-JOTO and DDPG-JOTO show moderate variations, with TD3-JOTO slightly more stable than DDPG-JOTO. These results indicate that JATO-DRL maintains superior performance and robustness under diverse communication resource conditions.

Fig. 12 illustrates the Time-Energy Reward of different algorithms under varying numbers of USVs. Overall, the system reward tends to decrease with the number of USVs, as more USVs generate tasks

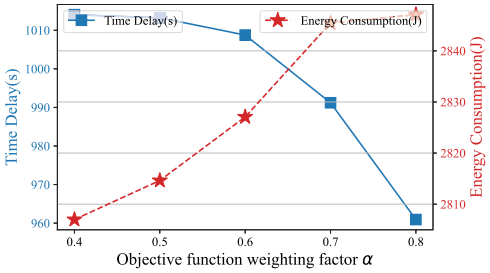


Fig. 9. Impact of weight factor on latency and energy consumption.

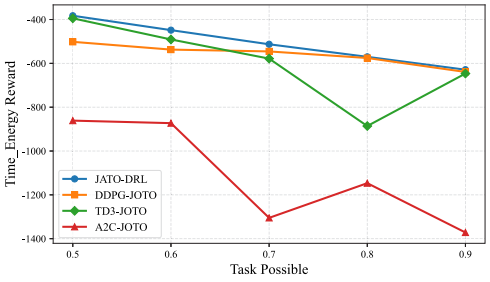


Fig. 10. System time-energy reward under varying task generation probabilities.

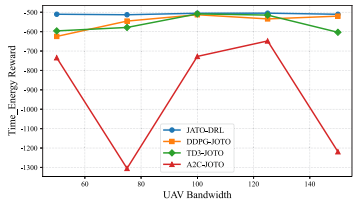


Fig. 11. System time-energy reward under varying bandwidth conditions.

simultaneously, intensifying competition for edge computing resources and communication bandwidth, which leads to higher task processing delays and energy consumption. JATO-DRL maintains the highest reward across most USV numbers (approximately −652 to −512), demonstrating strong robustness and efficient task scheduling. In contrast, A2C-JOTO and TD3-JOTO exhibit significant reward decreases for certain USV numbers, indicating reduced adaptability under high-load conditions, while DDPG-JOTO shows moderate performance but decrease notably when the number of USVs is large. These results indicate that JATO-DRL consistently achieves superior performance under different USV numbers, proving its reliability and efficiency in dynamic multi-USV task scenarios.

Fig. 13 shows the coverage achieved by different algorithms under varying numbers of USVs. Overall, JATO-DRL consistently achieves the highest coverage across all USV numbers (around 940–970), demonstrating its effectiveness in coordinating UAV to cover multiple USVs efficiently. In contrast, the baseline algorithms exhibit larger fluctuations: DDPG-JOTO and TD3-JOTO achieve moderate coverage with significant variability, while A2C-JOTO shows the lowest and most unstable coverage across different USV numbers. These results indicate that JATO-DRL not only maintains high coverage but also provides robust performance in dynamic multi-USV scenarios, effectively ensuring UAV can serve multiple USVs efficiently.

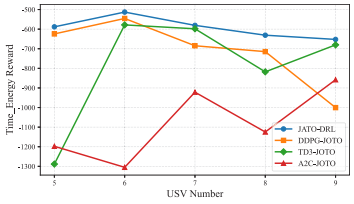


Fig. 12. System time-energy reward under varying numbers of USVs.

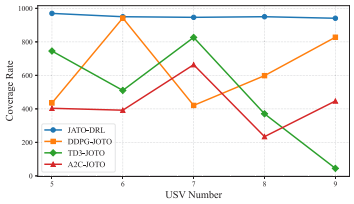


Fig. 13. Coverage of UAVs under varying numbers of USVs.

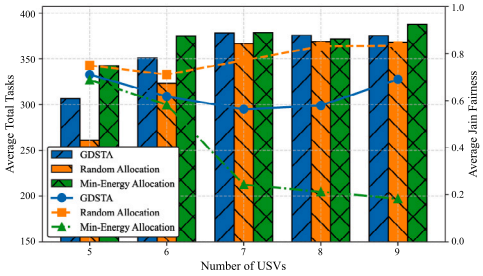


Fig. 14. Monitoring point accommodation and fairness under varying numbers of USVs.

Fig. 14 illustrates the number of monitoring points accommodated and the fairness of task allocation for different algorithms under varying numbers of USVs. Overall, the Random strategy achieves the most balanced distribution among USVs, indicating high fairness, but it results in the lowest total number of accommodated monitoring points. In contrast, the Min-Energy-Cost approach can achieve high total coverage but often leads to highly imbalanced allocations, with some USVs handling disproportionately many points. The GDSTA algorithm strikes a balance between the two, maintaining relatively high total coverage while keeping task allocation reasonably fair. These results indicate that GDSTA effectively balances coverage and fairness in dynamic multi-USV monitoring scenarios.

6.3. Practical considerations and Sim-to-Real gap

While the proposed framework is evaluated in a simulation environment, its deployment in real-world aquatic systems may introduce additional challenges, primarily due to the Sim-to-Real gap.

First, practical deployments are subject to limited computational resources and system constraints on embedded hardware platforms, which may lead to higher inference latency and computational overhead compared to simulation. Although the proposed DRL-based decision-making mechanism operates within the designed time slot in simulation, additional delays may arise in real systems due to processor limitations and scheduling overhead.

Second, real-world environments involve various uncertainties, including sensor noise, communication instability, and dynamic channel

conditions (e.g., multipath fading and interference), which are not fully captured in the current simulation model and may affect both task generation and offloading performance. Furthermore, hardware-related constraints such as power consumption limits and energy budgets of UAV and USVs may also impact overall system performance.

To address these challenges, the proposed framework is designed with strong engineering extensibility. In particular, model compression and quantization techniques can be employed to reduce inference latency, while hardware-aware optimization can improve execution efficiency on embedded platforms. In addition, incorporating more realistic environmental modeling and uncertainty-aware mechanisms can further enhance system robustness. Strategies such as partial task offloading and adaptive resource allocation also provide effective means to maintain real-time performance under practical constraints.

It is worth noting that incorporating detailed real-world factors (e.g., fine-grained channel modeling, sensor noise characterization, and hardware-specific constraints) requires extensive system-level integration and is beyond the primary scope of this work, which focuses on the design and validation of the decision-making framework. These aspects will be considered in future work.

Overall, the proposed framework has been systematically validated at the simulation and system-design level in terms of task scheduling, resource allocation, and multi-agent coordination. The adopted time-slot modeling, task offloading mechanism, and resource management strategies are consistent with current edge computing system design paradigms, indicating its potential practical feasibility and providing valuable insights for real-world aquatic monitoring system deployment.

7. Conclusion

In this paper, we propose an intelligent system integrating multi-USV path planning and UAV-assisted edge computing, leveraging DRL-based strategies for efficient and fair aquatic monitoring. Simulation results demonstrate that the proposed JATO-DRL and GDSTA methods outperform baseline algorithms in terms of efficiency, robustness, and coverage. Future work will focus on deploying the system in real-world aquatic environments to enable large-scale multi-agent collaboration and robust performance under dynamic conditions, as well as extending the current time-slotted model to support more flexible and asynchronous operations.

CRediT authorship contribution statement

Chaogang Tang: Methodology, Investigation. **Tiyu Yao:** Writing – original draft, Data curation, Conceptualization. **Shuo Xiao:** Writing – review & editing, Validation, Supervision. **Huaming Wu:** Investigation, Formal analysis. **Ruidong Li:** Validation, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 62476276 and No. 62271486), the Emerging Frontiers Cultivation Program of Tianjin University Interdisciplinary Center, Tianjin Natural Science Foundation Project (No. 25JCYBJC01540), and Tianjin Municipal Science and Technology Major Program (No. 25ZXZSSS00390).

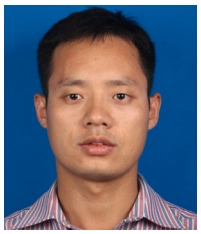
Data availability

Data will be made available on request.

References

[1] B. Chen, M. Wang, M. Duan, X. Ma, J. Hong, F. Xie, R. Zhang, X. Li, In search of key: Protecting human health and the ecosystem from water pollution in China, J. Clean. Prod. 228 (2019) 101–111.
[2] S.R. Carpenter, Eutrophication of aquatic ecosystems: bistability and soil phosphorus, Proc. Natl. Acad. Sci. 102 (29) (2005) 10002–10005.
[3] A. Hangan, C.-G. Chiru, D. Arsene, Z. Czako, D.F. Lisman, M. Mocanu, B. Pahontu, A. Predescu, G. Sebestyen, Advanced techniques for monitoring and management of urban water infrastructures—An overview, Water 14 (14) (2022) 2174.
[4] Y. Chen, D. Han, Water quality monitoring in smart city: A pilot project, Autom. Constr. 89 (2018) 307–316.
[5] A. Dauda, O. Flauzac, F. Nolot, A survey on IoT application architectures, Sensors 24 (16) (2024) 5320.
[6] J.E. Manley, D. Puzzuoli, C. Taylor, F. Stahr, J. Angus, A new approach to multi-domain ocean monitoring: Combining UAS with USVs, in: OCEANS 2025 Brest, IEEE, 2025, pp. 1–5.
[7] X. Liu, L. Ho, S. Bruneel, P. Goethals, Applications of unmanned vehicle systems for multi-spatial scale monitoring and management of aquatic ecosystems: A review, Ecol. Informatics (2024) 102926.
[8] Y. Fang, Z. Kuang, H. Wang, S. Lin, A. Liu, Minimizing energy consumption of collaborative deployment and task offloading in two-tier UAV edge computing networks, J. Syst. Archit. 167 (2025) 103511.
[9] Z. Liu, Y. Zhang, X. Yu, C. Yuan, Unmanned surface vehicles: An overview of developments and challenges, Annu. Rev. Control. 41 (2016) 71–93.
[10] M.J. Er, C. Ma, T. Liu, H. Gong, Intelligent motion control of unmanned surface vehicles: A critical review, Ocean Eng. 280 (2023) 114562.
[11] Y. Toyomoto, K. Oishi, T. Oshima, T. Hatanaka, Combined straight and circular path generation for aquatic environmental monitoring with multiple USVs, Adv. Robot. 39 (16) (2025) 1000–1013.
[12] T. Oshima, Y. Toyomoto, T. Hatanaka, Online coverage path generation for aquatic environmental monitoring with multiple USVs through constraint-based control, in: 2024 IEEE Conference on Control Technology and Applications, CCTA, IEEE, 2024, pp. 294–301.
[13] L. Zhao, Y. Bai, J.K. Paik, Global path planning and waypoint following for heterogeneous unmanned surface vehicles assisting inland water monitoring, J. Ocean. Eng. Sci. (2023).
[14] Y. Huang, H. Zhang, J. Liu, P. Li, J. Ding, D. Xu, Nonlinear ocean energy harvesting method for unmanned surface vessels, Renew. Energy (2025) 123383.
[15] N. Su, J.-B. Wang, C. Zeng, H. Zhang, M. Lin, G.Y. Li, Unmanned-surface-vehicle-aided maritime data collection using deep reinforcement learning, IEEE Internet Things J. 9 (20) (2022) 19773–19786.
[16] C. Li, C. Deng, Y. Zhang, S. Wan, Federated meta-learning based computation offloading approach with energy-delay tradeoffs in UAV-assisted VEC, IEEE Trans. Mob. Comput. 24 (10) (2025) 10978–10991.
[17] C. Li, Z. Zhang, J. Wang, S. Yuan, Z. Wang, L. Chai, A. Li, S. Wan, Improved multi-agent proximal policy optimization algorithm for resource allocation with radar-perception in UAV-assisted VEC, ACM Trans. Sens. Networks 21 (6) (2025) 58:1–58:28.
[18] C. Tang, Y. Ding, S. Xiao, Z. Huang, H. Wu, Collaborative service caching, task offloading, and resource allocation in caching-assisted mobile edge computing, IEEE Trans. Serv. Comput. 18 (4) (2025) 1966–1981.
[19] P. Yang, X. Deng, L. Wang, S. Lin, J. Gui, X. Chen, S. Wan, Y. Qian, Energy-efficient symbiotic UAV-enabled MEC networks via RIS: joint trajectory and phase-shift control optimization, IEEE Trans. Intell. Transp. Syst. 25 (12) (2024) 21142–21156.
[20] Y. Wang, H. Wang, J. Ding, H. Yu, From latency bottlenecks to seamless edge: AD3PG-powered joint optimization of UAV trajectory and task offloading, Comput. Netw. (2025) 111700.
[21] Z. Tong, X. Deng, Y. Zhang, J. Mei, C. Wang, K. Li, MADDPG-based task offloading and resource pricing in edge collaboration environment, J. Syst. Archit. 165 (2025) 103433.
[22] L. Zhao, Y. Yao, J. Guo, Q. Zuo, V.C. Leung, Collaborative computation offloading and wireless charging scheduling in multi-UAV-assisted MEC networks: A TD3-based approach, Comput. Netw. 251 (2024) 110615.
[23] Z. Wang, H. Wang, L. Liu, E. Sun, H. Zhang, Z. Li, C. Fang, M. Li, Dynamic trajectory design for multi-UAV-assisted mobile edge computing, IEEE Trans. Veh. Technol. (2024).
[24] S. Yuan, Q. Zhou, J. Li, S. Guo, H. Chen, C. Wu, Y. Yang, Adaptive incentive and resource allocation for blockchain-supported edge video streaming systems: A cooperative learning approach, IEEE Trans. Mob. Comput. (2024).
[25] S. Yuan, J. Li, C. Wu, JORA: Blockchain-based efficient joint computing offloading and resource allocation for edge video streaming systems, J. Syst. Archit. 133 (2022) 102740.
[26] A.M. Seid, G.O. Boateng, B. Mareri, G. Sun, W. Jiang, Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4531–4547.

[27] Z. Zhang, L. Zhu, A review on unmanned aerial vehicle remote sensing: Platforms, sensors, data processing methods, and applications, Drones 7 (6) (2023) 398.
[28] G. Katsouras, E. Dimitriou, S. Karavoltsos, S. Samios, A. Sakellari, A. Mentzafou, N. Tsalas, M. Scoullos, Use of unmanned surface vehicles (USVs) in water chemistry studies, Sensors 24 (9) (2024) 2809.

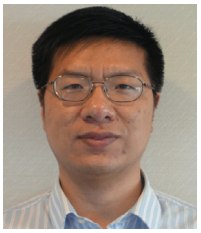


Chaogang Tang (Member, IEEE) received his B.S. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, and Ph.D. degree from the School of Information Science and Technology, University of Science and Technology of China, Hefei, China, and the Department of Computer Science, City University of Hong Kong, under a joint Ph.D. Program, in 2012. He is now with the China University of Mining and Technology. His research interests include mobile cloud computing, fog computing, Internet of Things, and big data.

Tiyu Yao is currently pursuing the M.S. degree with the China University of Mining and Technology, China. His research interests include mobile edge computing, deep learning, and multi-agent reinforcement learning.

Shuo Xiao received the Ph.D. degree in traffic information engineering and control from Beijing Jiaotong University in 2010. He has worked at China University of Mining and Technology since 2010, where he is now a professor. His research interests include the Internet of Things and measurement systems.

Huaming Wu (Senior Member, IEEE) received the B.E. and M.S. degrees from Harbin Institute of Technology, China in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universität Berlin, Germany in 2015. He is currently a professor at the Center for Applied Mathematics, Tianjin University, China. His research interests include mobile cloud computing, edge computing, Internet of Things, deep learning, complex networks, and DNA storage.



Ruidong Li is an associate professor at Kanazawa University, Japan. Before joining this university, he was a senior researcher at the National Institute of Information and Communications Technology (NICT), Japan. He received the M.Sc. degree and Ph.D. degree in computer science from the University of Tsukuba in 2005 and 2008, respectively. He serves as the secretary of IEEE ComSoc Internet Technical Committee (ITC), and are the founders and chairs of IEEE SIG on Big Data Intelligent Networking and IEEE SIG on Intelligent Internet Edge. He is the associate editor of IEEE Internet of Things Journal, and also served as the guest editors for a set of prestigious magazines, transactions, and journals, such as IEEE Communications Magazine, IEEE Network, IEEE TNSE. He also served as chairs for several conferences and workshops, such as the general co-chair for IEEE MSN 2021, AIVR2019, IEEE INFOCOM 2019/2020/2021 ICCN workshop, TPC co-chair for IWQoS 2021, IEEE MSN 2020, BRAINS 2020, IEEE ICDCS 2019/2020 NMIC workshop, and ICCSSE 2019. His research interests include future networks, big data, intelligent Internet edge, Internet of things, network security, information-centric network, artificial intelligence, quantum Internet, cyber–physical system, and wireless networks. He is a senior member of IEEE and a member of IEICE.