Energy-Aware USV-UAV Cooperative Task Offloading Optimization in Water Monitoring System

Chaogang Tang*, Tiyu Yao*, Shuo Xiao*, Huaming Wu†, Ruidong Li‡
*School of Computer Science and Technology / School of Artificial Intelligence,
China University of Mining and Technology, 221116, Xuzhou, China
†The Center for Applied Mathematics, Tianjin University, 300072, Tianjin, China
‡The Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan
{cgtang, ts24170138p31, sxiao}@cumt.edu.cn, whming@tju.edu.cn, liruidong@ieee.org

2026 IEEE 46th International Conference on Distributed Computing Systems (ICDCS) | 979-8-3195-2979-4/26/$31.00 ©2026 IEEE | DOI: 10.1109/2575-8411.2026.00080

***Abstract*—Traditional underwater monitoring systems often suffer from high latency and excessive energy consumption when transmitting sensor data to the cloud for processing. To address these limitations, we propose an intelligent water monitoring framework that integrates Mobile Crowd Sensing (MCS) and Mobile Edge Computing (MEC). Unmanned Surface Vehicles (USVs) in this paradigm collect data from submerged sensors while navigating across water surfaces, and Unmanned Aerial Vehicles (UAVs) equipped with edge computing servers in the MEC architecture act as airborne computing nodes, facilitating real-time task offloading and data analysis. The objective of this paper is to minimize overall system delay and energy consumption through joint optimization of USV path planning, UAV-assisted task offloading, and computational resource allocation. To reduce the complexity, we decompose the problem into two interdependent subproblems: a USV path planning subproblem solved using an Ant Colony Optimization (ACO)-based approach, and a joint task offloading and resource allocation subproblem for UAVs addressed via a hybrid strategy combining deep reinforcement learning (DRL) and Sequential Least Squares Programming (SLSQP). The simulation results show that our approach outperforms existing methods in terms of both feasibility and effectiveness, achieving significantly lower latency and energy usage in smart water monitoring applications.**

***Index Terms*—Mobile edge computing, smart cities, task offloading, path planning, resource allocation, deep reinforcement learning**

I. Introduction

With the continuous development of smart cities, the ecological monitoring of urban water bodies such as lakes has become increasingly important, serving as a key component in achieving green and sustainable urban development as well as ecological security governance [1]. Multidimensional aquatic information, such as fish behavior and fluctuations in water quality, requires more timely and accurate monitoring. Currently, the majority of underwater sensors gather aquatic data, which is then transmitted to cloud servers for centralized processing [2]. Due to the inconsistency between underwater and above-water communication media, such an approach often fails to meet the requirements of efficient monitoring and intelligent response [3].

In recent years, the integration of Unmanned Surface Vehicles (USVs) with Mobile Edge Computing (MEC) has become a prominent research focus [4]. Owing to the advantages including convenient deployment and adaptable perception [5], USVs are capable of autonomous surface navigation, patrolling, and environmental monitoring while operating on water bodies such as lakes. UAV assisted MEC deploys computing resources at the network edge, thus enabling low-latency and efficient data processing closer to data sources.

Many studies have paid attention to the multi-UAV cooperative offloading and resource allocation in USV-asssited MEC scenarios [6], [7]. Although these studies provide a theoretical foundation and technical support for the development of intelligent aquatic monitoring systems, there remain two key challenges: 1) How to achieve efficient USV path planning and communication coverage under energy constraints; 2) How to make task offloading and computing resource allocation decisions among multiple UAVs to minimize task processing delay and overall system energy consumption, while considering multiple constraint conditions.

To address the aforementioned challenges, we propose a USV-UAV cooperative intelligent aquatic monitoring system that integrates MCS with MEC. The system consists of two distinct entities: USVs and UAVs, respectively. USVs are responsible for dynamically collecting data from underwater sensors. A critical issue thus arises: how to efficiently assign cruise missions to a limited number of USVs for data collection while minimizing energy consumption. UAVs on the other hand can assist in handling computation tasks offloaded from USVs. The concern associated with UAVs is how to determine offloading destinations from USVs to UAVs and resource allocations among UAVs for offloaded tasks. In this paper, we investigate these issues and propose efficient algorithms to solve them.

The major contributions of this paper are summarized as follows:

- A USV-UAV collaborative underwater sensing and computation offloading system is put forward to support efficient collection and processing of multi-source hetero-

979-8-3195-2979-4/26/$31.00 ©2026 IEEE
DOI 10.1109/2575-8411.2026.00080

geneous aquatic data, by combining the merits of MCS and MEC.
- We aim to minimize the total delay and energy consumption by optimizing path planning, task offloading, and resource allocation in the system. To reduce the difficulty in solving it, we decompose it into several subproblems, with each to be tackled by different algorithms.
- We put forward an energy-aware path planning strategy based on K-means and Ant Colony Optimization (KACO) algorithms for USVs while cruising and performing MCS tasks on the lake. In addition, a DRL-based offloading algorithm is adopted to make offloading decisions from USVs to UAVs. Last but not least, the proposed SLSQP algorithm is employed to schedule computing resources for the offloaded tasks.
- Extensive simulation were conducted under various scenarios and the results demonstrate that the proposed strategy in this paper is better than the baseline approaches regarding all tested conditions.

II. RELATED WORK

MCS, as a distributed environmental perception paradigm, has been widely applied to urban environments and aquatic monitoring tasks [8], [9]. The core idea lies in leveraging mobile agents to replace static sensor nodes, thereby dynamically collecting environmental data to enhance sensing coverage and flexibility. For instance, McMahon et al. [10] proposed a multi-layer planning-based collaborative framework in which USVs dynamically detect target points and collaborate with Autonomous Underwater Vehicles (AUVs) to perform multi-target path planning. Cheng et al. [11] investigated the deployment strategies of USVs and AUVs as mobile receivers and relay nodes from the perspective of communication reliability, enabling robust data acquisition from both shallow and deep-water sensors while significantly reducing the bit error rate and enhancing transmission stability. Zhao et al. [12] further explored the autonomous data collection capability of USVs in complex and unknown water environments.

Path planning is a key component in the execution of sensing tasks, typically aimed at minimizing the total path length. For example, Liu et al. [13] proposed an improved ACO algorithm for collaborative multi-agent path optimization in coverage tasks. Zhuang et al. [14] addressed the dynamic scheduling of heterogeneous marine tasks by proposing a hybrid offline-online multi-agent task assignment and path planning framework. Furthermore, to address the limitations of traditional GA in multi-agent scenarios. Liu et al. [15] introduced an asynchronous genetic algorithm (AGA) to enhance planning responsiveness and stability. In terms of coverage planning, Bahabadi et al. [16] developed a "rendezvous–coverage" two-stage collaborative framework. The method performs Voronoi division using a triangular mesh and jointly applies ACO and GA for multi-agent path optimization, improving visiting order efficiency. Yin et al. [17] proposed an iso-time waypoint planning method that generates time-synchronized waypoints in free space to construct path sequences.

Recently, researchers have increasingly utilized UAVs as aerial edge nodes to assist computation [18]. Tang et al. [19] proposed a multi-UAV cooperative federated learning framework that addresses the privacy and local training limitations of traditional AIoT devices. Falcão et al. [20] established an energy-aware Markov chain model for virtual resource adaptation in NFV-MEC systems. By integrating node availability modeling with genetic algorithm-driven UAV resource allocation, their framework achieved 44% energy savings and guaranteed URLLC-grade service availability. Liao et al. [21] proposed a cooperative UAV-USV MEC platform where UAVs act both as communication relays and edge computing nodes to assist USVs in data computation and forwarding. They jointly optimized USV task execution modes, UAV trajectories, arrival times and hovering positions, forming an energy minimization problem.

Although significant progress has been made in path planning and computation offloading, there remain two main shortcomings: (1) Path planning and task offloading are mostly designed independently, lacking integrated coordination of perception and computing. (2) In complex heterogeneous multi-agent systems, there is a lack of joint optimization mechanisms that simultaneously consider energy balance, task coverage and response latency. To address these gaps, we propose a combined KACO path planning and MAPPO-CO-based task offloading, followed by UAV computational resource allocation optimized using the SLSQP algorithm, achieving efficient multi-USV and multi-UAV collaboration and resource allocation in smart aquatic scenarios, thereby improving overall perception quality and computing efficiency.

III. SYSTEM MODEL

As illustrated in Fig. 1, we consider an intelligent water environment monitoring system composed of S sensors, M USVs and N UAVs. The sets of sensors, USVs and UAVs are denoted by $\mathcal{S} = \{1, \ldots, S\}$, $\mathcal{M} = \{1, \ldots, M\}$, and $\mathcal{N} = \{1, \ldots, N\}$, respectively. In this paper, Time Division Multiple Access (TDMA) mechanism is adopted to divide continuous time into equal-length time slots, denoted as $\mathcal{T} = \{1, \ldots, T\}$. A total of S sensors are assumed to be randomly deployed underwater. The communication capability of each USV is determined by a maximum communication radius $D_{\max}^{\text{USV}} = h \tan \theta_{\text{USV}}$, where θ_{USV} denotes the maximum elevation angle of the USV and h is the water depth. The USV is further constrained by its maximum simultaneous task reception capacity, denoted by $num_{\max}$.

Assume that S sensors are partitioned into multiple clusters, according to the geographical locations of sensors. We regard each cluster center as one monitoring point, and all the sensors in one cluster can be served by the same USV. USV can be assigned to visit different monitoring points and collect aquatic tasks from the sensors. Given the overall stability of the urban lake environment, we ignore the effect of water speed. When a USV arrives at a monitoring point, it can maintain a hovering state simply by ceasing propulsion, and will resume its journey to the next monitoring point once the task is completed. Sensor

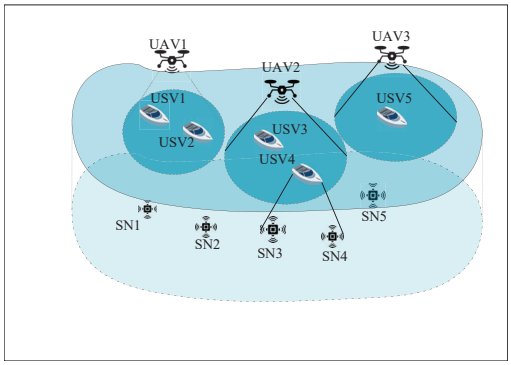


Fig. 1: Intelligent water monitoring system

task W_s is represented as a triple $(D_s, C_s, level_s)$, where D_s denotes the data size, C_s represents the required CPU cycles for accomplishing the task and $level_s \in \{1, \ldots, L\}$ indicates the task priority level. This priority determines the execution order of each task W_s on the USVs. We assume that each sensor is associated with one task and USVs have built-in dynamic obstacle avoidance capabilities. To simplify the model, the energy consumption for obstacle avoidance is ignored in this paper. Each UAV can be deployed to serve the USVs within a specified subregion and hover at a fixed altitude H.

We adopt the Gauss-Markov random mobility model [22] to model the movement of one USV. Specifically, the speed of USV m at time t is updated as:

$$V_m(t) = \alpha V_m(t-1) + (1-\alpha)\overline{V_m} + \sqrt{1-\alpha^2}\phi_m, \quad (1)$$

where $0 \leq \alpha \leq 1$ is a parameter used to adjust the influence of the previous state on the current state, $\overline{V_m}$ denotes the long-term average speed of USV m, and ϕ_m is a random variable that follows an independent Gaussian distribution.

A single USV can execute designated tasks at various monitoring points. Let P denote the global path planning decision composed of the trajectories of all USVs. For a given USV m, $P_m \in P$ represents its assigned trajectory set. Specifically, $\{P_m^a(t), P_m^b(t), P_m^c(t)\} \subseteq P_m$, representing the positions of the previous, current, and next monitoring points of USV m at time slot t. These variables are updated only when the USV has just arrived at a new monitoring point. Considering the slow cruising speed of the USV and the short duration of each time slot, we assume that the position coordinates are updated on a slot-by-slot basis. Therefore, the position of the moving USV m in the next time slot $t+1$ can be expressed as:

$$\begin{cases} x_m(t+1) = x_m(t) + \tau V_m(t)\cos\theta_m^{b,c}(t) \\ y_m(t+1) = y_m(t) + \tau V_m(t)\sin\theta_m^{b,c}(t) \end{cases} \quad (2)$$

where τ denotes the duration of each time slot. $\theta_m^{b,c}(t)$ denotes the direction of USV m from position $P_m^b(t)$ to the next position $P_m^c(t)$ at time slot t. Since the USV moves in straight lines between adjacent monitoring points, its direction only changes dynamically based on the positions of the current and next monitoring points, and,

$$\theta_m^{b,c}(t) = arctan\frac{y_m^c(t) - y_m^b(t)}{x_m^c(t) - x_m^b(t)}. \quad (3)$$

A. Mobility Energy Consumption Model for USVs

When the USV travels between two monitoring points, the turning energy consumption of USV m during a directional change, e.g., from its previous position $P_m^a(t)$, through the current position $P_m^b(t)$, to the subsequent position $P_m^c(t)$ at time slot t can be expressed as:

$$E_{m,a,b,c}^{turn}(t) = \kappa\theta_m^{a,b,c}(t)I_m^{b,c}(t), \quad (4)$$

where κ represents the comprehensive energy consumption coefficient, and $I_m^{b,c}(t)$ is the indicator function. If the USV m initiates its movement from the current monitoring point $P_m^b(t)$ to the next monitoring point $P_m^c(t)$ at time slot t, then $I_m^{b,c}(t) = 1$; otherwise, $I_m^{b,c}(t) = 0$. It should be noted that this indicator function is activated only in the first time slot of a movement. $\theta_m^{a,b,c}(t)$ denotes the turning angle of USV m, and,

$$\theta_m^{a,b,c}(t) = \arccos\left(\frac{\overrightarrow{ab(t)} \cdot \overrightarrow{bc(t)}}{\|\overrightarrow{ab(t)}\| \cdot \|\overrightarrow{bc(t)}\|}\right), \quad (5)$$

where $\overrightarrow{ab(t)} = (x_m^b(t) - x_m^a(t), y_m^b(t) - y_m^a(t))$ and $\overrightarrow{bc(t)} = (x_m^c(t) - x_m^b(t), y_m^c(t) - y_m^b(t))$.

Generally, the water resistance encountered by the USV during navigation is considerably greater than the wind resistance. Therefore, we only consider the water resistance in this paper. According to the energy consumption model in [23], the resistance can be expressed as:

$$R[\overline{V_m(t)}] = \frac{1}{2}\rho_w C_f S_m \overline{V_m(t)}^2, \quad (6)$$

where ρ_w represents the density of water, C_f denotes the friction coefficient, and S_m is the wetted surface area of USV m. Therefore, the propulsion energy consumption per unit distance of USV m at time slot t is given by:

$$E_m^{unit}(t) = \frac{R\left[\overline{V_m(t)}\right]}{\eta_{prob}}, \quad (7)$$

where η_{prob} denotes the propulsion system efficiency of the USV.

The estimated energy consumption for USV m moving from monitoring point $P_m^b(t)$ to monitoring point $P_m^c(t)$ at time slot t is given by:

$$E_m^{mov}(t) = E_{m,a,b,c}^{turn}(t) + E_m^{unit}(t). \quad (8)$$

B. USV Local Computing Model

When the USV arrives at a monitoring point, it hovers to collect sensor tasks and execute computations. Considering the unique underwater channel properties and the shallow depth of urban lakes, blue-green light communication is adopted for data transmission between sensors and USVs [24]. Upon USV m reaching monitoring point $P_m^b(t)$ at time slot t, the distance

between underwater sensor s and USV m can be expressed as:

$$D_{s,m}^{IoT,USV}(t) = \sqrt{\left[x_s^{IoT} - x_m^{USV}(t)\right]^2 + \left[y_s^{IoT} - y_m^{USV}(t)\right]^2 + h^2}. \quad (9)$$

The channel gain is modeled as an exponential attenuation with respect to distance, which is a simplified form of the absorption loss described in the classical underwater acoustic propagation model [25]:

$$g_{s,m}^{IoT,USV}(t) = e^{-k_e D_{s,m}^{IoT,USV}(t)}, \quad (10)$$

where k_e represents the extinction coefficient of underwater optical propagation.

The orthogonal frequency division multiple access technology is adopted for communication between underwater sensors and USVs in this paper. Assume there are $num_m^{USV}(t)$ sensors connected to USV m at monitoring point $P_b^m(t)$ at time slot t, and then the bandwidth resources can be evenly allocated to sensor s by $B_{s,m}^{IoT}(t) = B_{USV}/num_m^{USV}(t)$, where B_{USV} denotes the total bandwidth resource available to a USV. The data transmission rate between sensor s and USV m is given by:

$$R_{s,m}^{IoT,USV}(t) = B_{s,m}^{IoT}(t)\log_2\left[1 + \frac{P_s^{IoT} g_{s,m}^{IoT,USV}(t)}{N_0 B_{s,m}^{IoT}(t)}\right], \quad (11)$$

where N_0 represents the noise power, and P_s^{IoT} denotes the transmission power of the sensor. Hence, the transmission delay of sensor s offloading task W_s to USV m is:

$$T_{s,m}^{IoT,USV}(t) = \frac{D_s(t)}{R_{s,m}^{IoT,USV}(t)}. \quad (12)$$

Upon receiving all tasks from underwater sensors, USV m sorts them according to task priority before performing unified computation. Therefore, at the monitoring point $P_b^m(t)$, the time from each sensor starting to upload its task until the task is scheduled by USV m can be uniformly expressed as:

$$T_{m,b}^{IoT,USV}(t) = \max_{W_s \in W_{P_b}} \left\{T_{s,m}^{IoT,USV}(t)\right\}, \quad (13)$$

where W_{P_b} denotes the unsorted task queue at monitoring point $P_b^m(t)$.

Assuming that the unsorted task queue W_{P_b} at monitoring point $P_b^m(t)$, after being sorted according to priority, is denoted as $W_b^{st} = \{W_1, \ldots, W_{num_m^{USV}(t)}\}$, $W_s \in W_b^{st}$, represents one task in the sorted queue. Then, the queuing delay for task W_{s+1} is:

$$T_{s+1}^{queue}(t) = T_s^{queue}(t) + T_s(t), \quad (14)$$

where $T_s(t)$ denotes the computation time of the previous task W_s, if W_s is the first task in the queue, then $T_s^{queue}(t) = T_{m,b}^{IoT,USV}(t)$.

For the discrete offloading decision, let $\mathcal{W}(t)$ denotes the task queue consisting of all pending tasks in the current time slot t. The offloading decision is represented as $X(t) = \{X_{s,m,n}(t) \mid W_s \in \mathcal{W}(t), 1 \le m \le M, n \in \mathcal{N}\}$. Here, $X_{s,m,n}(t) \in \{0,1\}$ is a binary variable, where $X_{s,m,n}(t) = 1$ indicates that task W_s from USV m is is offloaded to UAV n. Otherwise, $\sum_{n=1}^{N} X_{s,m,n}(t) = 0$, meaning that the task is not offloaded to any UAV and will thus be executed locally on the USV.

The local computation time for task W_s on USV m at time slot t is given by:

$$T_{m,n,s}^{com}(t) = \frac{\left[1 - \sum_{n=1}^{N} X_{s,m,n}(t)\right] D_s(t) C_s(t)}{f_{USV,m}}, \quad (15)$$

where $f_{USV,m}$ denotes the local computing capability of USV m. The response time for local computation of task W_s is:

$$T_{m,n,s}^{local}(t) = T_s^{queue}(t) + T_{m,n,s}^{com}(t). \quad (16)$$

The energy consumption of USV m for locally processing task W_s at time slot t is:

$$E_{m,n,s}^{local}(t) = \left[1 - \sum_{n=1}^{N} X_{s,m,n}(t)\right] k_{USV,m} {f_{USV,m}}^2 D_s(t) C_s(t), \quad (17)$$

where $k_{USV,m}$ denotes the effective switched capacitance of the CPU onboard USV m.

C. MEC Processing Model

Assuming that the UAV has a maximum elevation angle θ_{UAV}, the maximum horizontal radius for the UAV to connect with a USV is $D_{max}^{UAV} = H \tan\theta_{UAV}$. At time slot t, the distance between USV m and UAV n is:

$$D_{m,n}^{USV,UAV}(t) = \sqrt{\left[x_n^{UAV}(t) - x_m^{USV}(t)\right]^2 + \left[y_n^{UAV}(t) - y_m^{USV}(t)\right]^2 + H^2}. \quad (18)$$

Since there are very few obstacles when USVs on the lake communicate with UAVs in the air, the communication can be regarded as standard LOS. The free-space path loss between USV m and UAV n is [26]:

$$PL_{m,n}(t) = \left[\frac{4\pi D_{m,n}^{USV,UAV}(t)}{c}\right]^2, \quad (19)$$

where c denotes the speed of light. The channel gain between USV m and UAV n can be expressed as:

$$g_{m,n}^{USV,UAV}(t) = \frac{1}{PL_{m,n}(t)} = \left[\frac{c}{4\pi D_{m,n}^{USV,UAV}(t)}\right]^2. \quad (20)$$

Assume that the total bandwidth is evenly divided among the USVs connected to each UAV, and $num_n^{UAV}(t)$ is the number of USVs connected to UAV n at time slot t. Then, the bandwidth allocated to this USV is $B_m^{USV}(t) = B_{UAV}/num_n^{UAV}(t)$. The data transmission rate between USV m and UAV n is:

$$R_{m,n}^{USV,UAV}(t) = B_m^{USV}(t)\log_2\left[1 + \frac{P_{USV} g_{m,n}^{USV,UAV}(t)}{N_1 B_m^{USV}(t)}\right], \quad (21)$$

where N_1 represents the noise power between the USV and UAV, and P_{USV} denotes the transmission power of the USV. The transmission delay for offloading task W_s from USV m to UAV n is:

$$T_{m,n,s}^{USV,UAV}(t) = \frac{X_{s,m,n}(t)D_s(t)}{R_{m,n}^{USV,UAV}(t)}. \tag{22}$$

One UAV can process multiple tasks at the same time. Let $\mathcal{W}_n(t)$ denotes the task queue consisting of all tasks assigned to UAV n in the current time slot t, $f(t)$ denote the resource allocation strategy of all UAVs at time slot t, and $f(t) = \{f_{s,n}^{UAV}(t) | W_s \in \mathcal{W}_n(t), n \in \mathcal{N}\}$, where $f_{s,n}^{UAV}(t)$ is the computing resources allocated by UAV n to task W_s at time slot t, satisfying $\sum_{W_s \in \mathcal{W}_n(t)} f_{s,n}^{UAV}(t) \leq f_{UAV}$, where f_{UAV} represents the computing resources of UAV.

The computation delay for UAV n to process task W_s offloaded by USV m can be expressed as:

$$T_{m,n,s}^{com}(t) = \frac{X_{s,m,n}(t)D_s(t)C_s(t)}{f_{s,n}^{UAV}(t)}. \tag{23}$$

When the UAV finishes the task, it sends the results back to the USV. Considering the small size of output data, the return cost is negligible in this paper. Thus, the response time for task W_s, when USV m chooses to offload the computation to UAV n, actually includes the queuing time $T_s^{queue}(t)$, the offloading delay $T_{m,n,s}^{USV,UAV}(t)$, and the task execution time $T_{m,n,s}^{com}(t)$. That is,

$$T_{m,n,s}^{mec}(t) = T_{m,n,s}^{USV,UAV}(t) + T_s^{queue}(t) + T_{m,n,s}^{com}(t). \tag{24}$$

The energy consumption of USV m for offloading task W_s to UAV n is:

$$E_{m,n,s}^{post}(t) = X_{s,m,n}(t)P_{USV}T_{m,n,s}^{USV,UAV}(t). \tag{25}$$

The energy consumption of UAV m for receiving task W_s offloaded by USV n is:

$$E_{m,n,s}^{rec}(t) = X_{s,m,n}(t)P_{UAV}T_{m,n,s}^{USV,UAV}(t). \tag{26}$$

The energy consumption of UAV m for computing task W_s is:

$$E_{m,n,s}^{com}(t) = X_{s,m,n}(t)k_{UAV,n}f_{s,n}^{UAV}(t)^2 D_s(t)C_s(t), \tag{27}$$

where $k_{UAV,n}$ denotes the effective switched capacitance of the CPU onboard UAV n.

Therefore, the total energy consumption for task W_s at time slot t, when USV m chooses to offload computation to UAV n, can be expressed as:

$$E_{m,n,s}^{mec}(t) = E_{m,n,s}^{post}(t) + E_{m,n,s}^{rec}(t) + E_{m,n,s}^{com}(t). \tag{28}$$

IV. PROBLEM FORMULATION

We aim to minimize overall energy consumption and task response delay through the optimization of task offloading strategies and USV movement planning. Let $MU_m(t) \in \{0, 1\}$ denote the state of USV m at time slot t, where $MU_m(t) = 0$ indicates USV m is in motion and $MU_m(t) = 1$ represents that USV m keeps stationary at time slot t. The energy consumed by USV m for movement during time slot t is:

$$E_m^{move}(t) = \left[1 - MU_m(t)\right] E_m^{mov}(t). \tag{29}$$

The total delay to process task W_s in time slot t is:

$$T_s^{deal}(t) = MU_m(t)\left[T_{m,n,s}^{local}(t) + T_{m,n,s}^{mec}(t)\right], \tag{30}$$

and the total energy consumption is:

$$E_s^{deal}(t) = MU_m(t)\left[E_{m,n,s}^{local}(t) + E_{m,n,s}^{mec}(t)\right]. \tag{31}$$

Therefore, the overall utility function can be expressed as:

$$\eta(t) = \beta \sum_{m \in M} E_m^{\text{move}}(t) + \gamma \sum_{W_s \in \mathcal{W}(t)} T_s^{\text{deal}}(t) + \delta \sum_{W_s \in \mathcal{W}(t)} E_s^{\text{deal}}(t). \tag{32}$$

where β, γ, and δ are non-negative weight coefficients that balance the importance of movement energy, task delay, and computation energy, respectively.

Then, the optimization problem is formulated as:

$$J: \min_{P, X(t), f(t)} \sum_{t=1}^{T} \eta(t) \tag{33}$$

$$\text{s.t.} \sum_{n \in \mathcal{N}} X_{s,m,n}(t) \in \{0, 1\}, \forall W_s \in \mathcal{W}(t), m \in \mathcal{M}, \tag{34}$$

$$0 \leq x_m^{USV}(t) \leq x_{max}, \tag{35}$$

$$0 \leq y_m^{USV}(t) \leq y_{max}, \tag{36}$$

$$D_{m,n}^{USV,UAV}(t) \leq D_{max}^{UAV}, \quad \text{if } X_{s,m,n}(t) = 1, \tag{37}$$

$$0 \leq V_m(t) \leq V_{max}, \tag{38}$$

$$\sum_{W_s \in \mathcal{W}_n(t)} f_{s,n}^{UAV}(t) \leq f_{UAV}, \forall n, \tag{39}$$

$$0 \leq f_{s,n}^{UAV}(t), \tag{40}$$

where P denotes the global path planning decision of all USVs, $X(t)$ denotes the offloading decision and $f(t)$ indicates the resource allocation scheme.

The problem J is a mixed integer non-linear programming problem, which is NP-hard and non-convex. Solving both USV's positions and task-offloading decisions together requires exponential complexity. Since navigation energy dominates computation energy, the original problem can be decomposed into two subproblems, i.e., the subproblem J_1 for path planning, and the subproblem J_2 for task offloading and resource allocation. The subproblem J_1 takes the following form:

$$J_1: \min_{P} \sum_{t=1}^{T} \sum_{m \in \mathcal{M}} E_m^{move}(t) \tag{41}$$

$$\text{s.t. } (35),\ (36),\ (38) \tag{42}$$

Similarly, the subproblem J_2 can be formulated as:

$$J_2: \min_{X(t),f(t)} \sum_{t=1}^{T} \left[\omega \sum_{W_s \in \mathcal{W}(t)} \frac{T_s^{\text{deal}}(t)}{T_{\max}} + (1-\omega) \sum_{W_s \in \mathcal{W}(t)} \frac{E_s^{\text{deal}}(t)}{E_{\max}} \right] \quad \text{s.t. } (37),\ (39),\ (40) \tag{43}$$

The formula J_2 represents the joint optimization of response latency and energy consumption for task processing after the USV reaches a monitoring point. Here, $\omega \in [0,1]$ balances latency and energy. T_{max} and E_{max} are used for normalization to make the two objectives comparable.

V. ALGORITHM IMPLEMENTATION

For the optimization objective J_1, the KACO algorithm is employed to plan USV paths in order to minimize movement energy consumption. Regarding the optimization objective J_2, the decision variables include both discrete variables (i.e., task offloading decisions) and continuous variables (i.e., UAV allocated computing resources), making the overall problem non-convex. To tackle this, we propose the MAPPO-CO algorithm to determine task offloading strategies. Given a fixed offloading decision, the UAV resource allocation problem is convex with a linear constraint structure. We employ the SLSQP algorithm to efficiently obtain the optimal allocation.

A. KACO Path Planning Algorithm

The proposed KACO algorithm actually includes two phases. First, we adopt K-means clustering algorithm to classify the predefined monitoring points. Each cluster can be served by specified USV. Second, we put forward an energy-aware ACO strategy to optimize the paths of USVs within each cluster.

Specifically, all the monitoring points are divided into Q clusters based on the spatial distribution of these points. The estimated energy consumption for each cluster q can be expressed as:

$$E_q^{est} = E^{unit} \hat{L}_q + k \sum_{\theta \in \theta_q} \theta, \tag{44}$$

where E^{unit} represents the average estimated propulsion energy consumption coefficient across all USVs and $\hat{L}_q$ denotes the path length within the monitoring points cluster, and θ_q denotes the set of turning angles along the planned path.

In addition, we introduce a balancing factor $\alpha_{BC} \in [0,1]$ to achieve load balancing among clusters. The above balancing factor can be integrated into the following evaluation metric:

$$BalancedCost = \alpha_{BC} \sum_{q=1}^{Q} \left| n_q^{clu} - \bar{n} \right| + (1-\alpha_{BC}) \sum_{q=1}^{Q} E_q^{est}, \tag{45}$$

where n_q^{clu} represents the number of monitoring points in cluster q, and $\bar{n}$ denotes the average number of monitoring points. $BalancedCost$ is used to achieve the trade-off between the number of points and energy consumption.

Within each cluster, we use the ACO algorithm to construct a low-energy consumption visitation path, with the state transition probability defined as:

$$p_{i,j} = \frac{[\tau_{i,j}]^{\alpha} [\eta_{i,j}]^{\beta}}{\sum_{k \in allowed} [\tau_{i,k}]^{\alpha} [\eta_{i,k}]^{\beta}}, \eta_{i,j} = \frac{1}{d_{i,j}}, \tag{46}$$

where $\tau_{i,j}$ denotes the pheromone concentration on edge i,j, and $allowed$ represents the set of unvisited nodes that can be chosen as the next step from node i.

To further reduce the path energy consumption, a 2-opt local search optimization is applied to the path constructed by the ant colony. The core idea is to swap two segments of the path boundaries to achieve a lower-cost route. Each iteration tries to update by:

$$p^1 = p - \{(i,j),(k,l)\} + \{(i,k),(j,l)\}, \tag{47}$$

where p denotes the current path and p^1 represents the new path obtained by exchanging the two edges (i,j) and (k,l). If $E(p^1) < E(p)$, the adjustment is accepted. Then, in each iteration, the total energy consumption of all current path combinations is calculated as:

$$E_{total} = \sum_{q=1}^{Q} E_q^{ACO+2opt}, \tag{48}$$

and the standard ACO pheromone evaporation and reinforcement mechanisms are applied:

$$\tau_{i,j} \leftarrow (1-p)\tau_{i,j}, \tag{49}$$

$$\tau_{i,j} \leftarrow \tau_{i,j} + \Delta\tau_{i,j}, \quad \Delta\tau_{i,j} = C/E_{best}, \tag{50}$$

where E_{best} denotes the minimum energy consumption of a candidate path in ACO iterations, and C is a constant pheromone scaling factor to control the strength of pheromone deposition.

B. MAPPO-CO Offloading Decision Algorithm

For the task offloading decision, traditional algorithms struggle to obtain an optimal solution in the proposed multi-UAV-assisted USV edge computing scenario. Therefore, we put forward the MAPPO-CO algorithm to solve the problem. We formulate the optimization objective as a Multi-Agent MDP. The key components of the MDP include the state space S_t, action space A_t and reward function R_t.

State Space S_t: The environment's state contains all the information required for DRL and is composed of the states of UAVs and USVs. These states dynamically evolve over different time slots t. The state space can be represented as $S_t = \left\{ s_m^{USV}(t), s_n^{UAV}(t) \middle| m \in \{1,...,M\}, n \in \{1,...,N\} \right\}$.

The state of USV m is defined as $s_m^{USV}(t) = \left\{ f_m^{arrive}(t),\ d_m^{task}(t),\ c_m^{task}(t),\ f_m^{cpu}(t) \right\}$, with each element to denote the current operational state of USV m, the queue of task data sizes, the queue of execution cycles, and its computing capability, respectively. The state of UAV n relative

Algorithm 1: Multi-USV Cooperative Path Planning Based on KACO Algorithm

Input: Monitoring point set, number of USVs $\mathcal{M}$, USV speed list usv_speeds

Output: Total energy consumption $total\_energy$, USV paths P

Initialize parameters: $start\_point = [0, 0]$, ACO params (α, β, ρ, Q), number of ants A, max ACO iterations $MaxACOIter$, max clustering iterations $MaxClusteringIter$, $episode \leftarrow 0$, $P \leftarrow \emptyset$, $total\_energy \leftarrow 0$;

Compute energy coefficients R_m for each USV m;

while $episode < MaxEpisode$ **do**

  Initialize K-means clustering to divide monitoring points into $Q = M$ clusters;

  repeat

    Compute cluster energy potentials E_q^{est} using Eq. (44);

    Assign USVs to clusters according to energy coefficients R_m and cluster potentials;

    Perform load balancing: migrate task points between clusters using Eq. (45);

    Update cluster centroids and re-cluster task points;

  until *cluster balance condition is satisfied or max clustering iterations reached*;

  for *each USV* m **do**

    Initialize pheromone matrix $\tau_{i,j}$;

    for *each ACO iteration* k **do**

      for *each ant* a **do**

        Construct path p_m using Eq. (46);

        Compute path energy E_a using Eq. (44);

        Update best path p_m^{best};

      Update pheromone $\tau_{i,j}$ using Eq. (49) and Eq. (50);

    Refine path with 2-opt local search Eq. (47);

    Save best path p_m^{best} and energy E_m^{best} for USV m;

    Add p_m^{best} into path set P;

  Compute total energy $total\_energy$ using Eq. (48);

  $episode \leftarrow episode + 1$;

to USV m is given by $s_n^{UAV}(t) = \{d_n^{uav}(t),\ f_n^{uav}(t)\}$, with each element respectively representing the distance between the USVs within the communication range of UAV n and the UAV's computing capability.

Action Space A_t: In this paper, an action refers to the task offloading decision made by an agent based on the observed state. The action space can be represented as: $A_t = \{a_m(t) \mid m \in \mathcal{M}\}, a_m(t) \in \{\text{idle}\} \cup \{X(t) \mid W_s \in \mathcal{W}(t)\}$. Here, idle denotes the case where USV m does not have any task to process at time slot t.

Reward Function R_t: When an agent acts in a state, it receives an immediate reward. To minimize the weighted sum of delay and energy, the reward function is defined as $R_t = -\eta_2(t)$.

The core components of the MAPPO-CO algorithm include a policy network (Actor) and a value estimation network (Critic). The policy network Actor takes the environment state S_t as input. After two shared fully connected layers to extract common features, the network splits into M output heads, with each corresponding to one of the M USVs. Each head outputs a logits vector representing a multinomial distribution, from which an action is sampled as $a_t^m \sim \pi_{\theta^m}(a_t^m|S_t)$. The value network Critic also takes S_t as input and outputs a scalar value $V(S_t)$, which estimates the value of the current state to assist in policy evaluation and update. Similar to the PPO algorithm, MAPPO-CO also uses an advantage function to indicate the advantage of taking a certain action in the current state. It adopts the Generalized Advantage Estimation method to calculate the advantage A_t at each step to reduce high variance, as shown below:

$$\delta_t = r_t + \gamma V(S_{t+1}) - V(S_t), \tag{51}$$

$$\hat{A}_t = \sum_{ls=0}^{LS} (\gamma\lambda)^{ls} \delta_{(t+ls)}, \tag{52}$$

where δ_t is the Temporal Difference error, γ is the discount factor, λ controls the trade-off between bias and variance, and ls denotes the step index from the current time t in the summation. To prevent large distributional deviations between the new and old policies, the algorithm introduces a clipped objective function to optimize the policy gradient:

$$L^{CLIP}(\theta) = E_t\left[\min(ratio_t \hat{A}_t, clip(ratio_t, 1-\epsilon, 1+\epsilon)\hat{A}_t)\right], \tag{53}$$

where $\text{ratio}_t = \frac{\pi_\theta(a_t|S_t)}{\pi_{\theta_{\text{old}}}(a_t|S_t)}$ is the importance sampling ratio and ϵ is the clipping parameter used to prevent excessively rapid changes in the policy. The loss function of the policy network can be expressed as:

$$L^{\text{Critic}}(\theta') = E_t \delta_t^2. \tag{54}$$

C. UAV Computing Resource Allocation under Fixed Offloading Decisions

After the task offloading decisions have been determined using the MAPPO-CO algorithm and the tasks have been uploaded from the USVs to the UAVs, the next critical step is to efficiently allocate the limited computational resources of each UAV among its assigned tasks. Let $f_n(t) = \{f_{s,n}^{\text{UAV}}(t) \mid W_s \in \mathcal{W}_n(t)\}$, where $f_{s,n}^{\text{UAV}}(t) > 0$ for $\forall W_s \in \mathcal{W}_n(t)$,

denoting the computation resource allocation set of UAV n at time slot t. The slot-level optimization for UAV n is:

$$\min_{f_n(t)} G(f_n(t)) = \sum_{W_s \in \mathcal{W}_n(t)} g_s(f_{s,n}^{\mathrm{UAV}}(t))$$

$$= \sum_{W_s \in \mathcal{W}_n(t)} \left[\gamma \frac{D_s C_s}{f_{s,n}^{\mathrm{UAV}}(t)} + \delta k_n D_s C_s \big(f_{s,n}^{\mathrm{UAV}}(t)\big)^2 \right] \quad (55)$$

$$\text{s.t.} \quad \sum_{W_s \in \mathcal{W}_n(t)} f_{s,n}^{\mathrm{UAV}}(t) \le f_{\mathrm{UAV},n}, \quad (56)$$

$$f_{s,n}^{\mathrm{UAV}}(t) > 0, \quad \forall W_s \in \mathcal{W}_n(t). \quad (57)$$

We now show that this is a convex optimization problem. First, the objective function is separable with respect to each task W_s, i.e.,

$$g_s(f) = \gamma \frac{D_s C_s}{f} + \delta k D_s C_s f^2, \quad (58)$$

where $f = f_{s,n}^{UAV}(t) > 0$ since the task has been offloaded to UAV n.

The first-order derivative of $g_s(f)$ with respect to f is:

$$\frac{\partial g_s}{\partial f} = -\gamma \frac{D_s C_s}{f^2} + 2\delta k D_s C_s f. \quad (59)$$

The second-order derivative is:

$$\frac{\partial^2 g_s}{\partial f^2} = 2\gamma \frac{D_s C_s}{f^3} + 2\delta k D_s C_s. \quad (60)$$

which is strictly positive for all $f > 0$. Therefore, $g_s(f)$ is strictly convex. Since the objective function is the sum of the costs of all tasks, we can express it in vector form as:

$$\nabla G(f_n(t)) = \begin{bmatrix} \frac{\partial g_{s_1}}{\partial f_{s_1}} \\ \frac{\partial g_{s_2}}{\partial f_{s_2}} \\ \vdots \end{bmatrix}, \quad (61)$$

$$\nabla^2 G(f_n(t)) = \mathrm{diag}\left(\frac{\partial^2 g_s}{\partial f_s^2} \right)_{W_s \in \mathcal{W}_n(t)}. \quad (62)$$

Since the Hessian matrix is diagonal with strictly positive diagonal elements, $\nabla^2 G$ is positive definite, which implies that G is strictly convex with respect to $f_n(t)$. The constraint (56) is linear, and (57) defines a convex domain. Therefore, problem (55)–(57) is a convex optimization problem and admits a unique global optimal solution.

To efficiently compute the optimal allocation, we employ the SLSQP algorithm to solve for $f_n(t)$. The procedure begins with an initialization phase, where a feasible allocation $f_n(t) = \{f_{s,n}^{\mathrm{UAV}}(t) \mid W_s \in \mathcal{W}_n(t)\}$ is selected such that $\sum_{W_s \in \mathcal{W}_n(t)} f_{s,n}^{\mathrm{UAV}}(t) \le f_{UAV,n}$ and $f_{s,n}^{\mathrm{UAV}}(t) > 0$ holds for all tasks.

Based on this initial point, the gradient and Hessian of the vectorized objective $G(f_n(t))$ are computed, as shown in (61) and (62). Using these differential quantities, the algorithm constructs a local quadratic approximation of the objective. At iteration k, the search direction is obtained by solving the following quadratic subproblem:

$$\min_d \ \nabla G(f^k)^T d + \tfrac{1}{2} d^T \nabla^2 G(f^k) d, \quad (63)$$

subject to the linearized constraints

$$\sum_{W_s \in \mathcal{W}_n(t)} (f_{s,n}^{\mathrm{UAV}}(t) + d_s) \le f_{UAV,n}, \qquad f_{s,n}^{\mathrm{UAV}}(t) + d_s > 0. \quad (64)$$

Once the search direction d^k is obtained, the allocation is updated according to

$$f^{k+1} = f^k + \alpha d^k, \qquad 0 < \alpha \le 1, \quad (65)$$

where the step size α is determined through a line-search procedure. This iterative refinement continues until the gradient norm $\|\nabla G(f)\|$ falls below a prescribed threshold or the allocation updates become negligible.

Because the objective is strictly convex and all constraints are linear, the SLSQP procedure is guaranteed to converge to the unique global optimum of the allocation problem.

VI. SIMULATION EXPERIMENTS AND RESULT ANALYSIS

A. Simulation Setups

The experimental environment is based on the Python 3.10 platform with the PyTorch 2.0.1 framework. The simulated area is a 400 × 400 m^2 water surface, containing 500–1500 underwater sensors, 5–25 USVs, 3 UAVs, and a water depth of 50 m. To ensure reliable communication and task offloading, the UAV positions are pre-determined to maximize the overall spatial coverage. Specifically, UAV locations are initialized based on the spatial distribution of sensor cluster centers and are slightly adjusted to minimize overlap among coverage regions. As shown in Fig. 2, the coverage regions of the three UAVs collectively span the vast majority of the monitoring points. This indicates that nearly all monitoring points lie within at least one UAV's coverage area, ensuring that the USVs remain within UAV communication range when performing tasks at these locations. In the proposed MAPPO-CO algorithm, the actor network is configured with a learning rate of 0.0001. Its architecture consists of two shared hidden layers with 128 and 256 neurons, respectively, followed by independent output heads for each agent. Each output head contains two fully connected layers with 128 neurons and produces 5 action outputs. The critic network adopts a learning rate of 0.001 and consists of three layers containing 128, 256, and 1 neuron, respectively. Additional simulation parameters related to communication, tasks, and vehicle characteristics are summarized in Table I.

B. Baseline Approaches

In terms of USV path planning, we introduce four baseline path planning algorithms for comparative analysis:

- *ACO algorithm*: The ant colony algorithm mimics ant foraging, where agents explore paths and update pheromones to guide future choices. Through iterative optimization

Algorithm 2: Joint Task Offloading and UAV Resource Allocation via MAPPO-CO and SLSQP

Input: Maximum training epochs $\boldsymbol{MT}$, PPO update rounds $\boldsymbol{N}$, update frequency $\boldsymbol{U}$

Output: Offloading decisions $\{\boldsymbol{A_t}\}$, resource allocation strategy $\{f(t)\}$

Initialize policy network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$, experience replay buffer $\boldsymbol{D}$, set iteration counter $\boldsymbol{t \leftarrow 0}$;

while $t < MT$ **do**

  Initialize environment, obtain initial system state $\boldsymbol{S_t}$;

  while *not all tasks are completed* **do**

    Select action $\boldsymbol{A_t}$ (offloading decisions for each USV) based on current policy π_t;

    Execute action $\boldsymbol{A_t}$, tasks are offloaded to UAVs;

    Solve the convex UAV resource allocation problem using SLSQP to obtain $\{f(t)\}$;

    Perform task computation on UAVs using allocated resources;

    Compute reward $\boldsymbol{R_t}$ based on execution delay and energy consumption;

    Store $(S_t, A_t, R_t, S_{t+1}, \text{done})$ in buffer $\boldsymbol{D}$;

    if *number of collected experiences* $\geq U$ **then**

      Compute advantage estimates $\hat{\boldsymbol{A}}_{\boldsymbol{t}}$ using Eq. (52);

      for $n = 1$ **to** N **do**

        Sample mini-batch from $\boldsymbol{D}$;

        Construct probability ratios between current and old policies;

        Update Actor parameters $\boldsymbol{\theta}$ using Eq. (53);

        Update Critic parameters $\boldsymbol{\theta'}$ using Eq. (54);

        Add entropy regularization to encourage exploration;

  $t \leftarrow t + 1$;

and collaboration, it seeks the path with minimum energy consumption.

- *Greedy Algorithm (Greedy)*: Greedy algorithm chooses the locally optimal option with the least cost or the highest benefit at each step without considering future consequences. Its objective is to build a complete path solution through a series of stepwise optimal decisions.
- *Hungarian Algorithm (Munkres)*: Munkres solves the cost matrix to minimize the total assignment cost between agents and tasks [27]. In path planning, it can optimally assign multiple agents to target points in one step to achieve minimal overall energy consumption.
- *Genetic algorithm (GA)*: GA is a global optimization method inspired by natural selection. It encodes paths as individuals and iteratively improves the population via selection, crossover, and mutation to approach an optimal solution. In path planning, GA can find coordinated optimal paths for multiple agents in a large search space, minimizing total energy consumption.

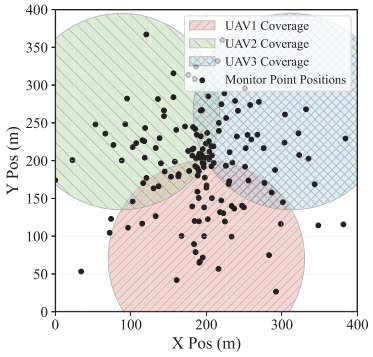


Fig. 2: UAV Coverage Distribution and Monitor Point Positions

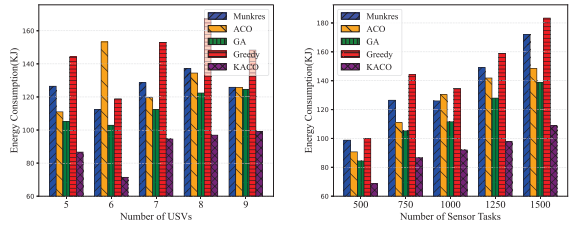
TABLE I: Simulation Parameters

Parameter Description	Value
Sensor Task Size	1–3 Mb
Sensor Task Cycle Frequency	200–360 MHz
Wireless Bandwidth	5 MHz
USV Speed	0–3 m/s
USV Computing Capacity	1.5–2 GHz
UAV Computing Capacity	15–35 GHz
Maximum Elevation Angle	60°
USV Upload Power	0.1 W
UAV Receiving Power	0.1 W
UAV flight altitude	75m
Gaussian White Noise PSD	-107 dBm
Underwater Extinction Coefficient	0.20

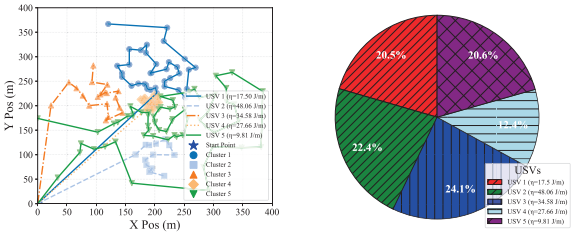
In the meanwhile, in terms of task execution, we also introduce three baseline approaches for performance comparison:

- *Random Offloading Computation Offloading Algorithm (ROCO)*: ROCO assigns each offloadable task randomly to an available node for computation without considering task characteristics or network status.
- *A2C-based Computation Offloading Algorithm (A2CCO)*: A2CCO leverages the A2C reinforcement learning framework, where the actor generates offloading actions and the critic network evaluates them based on system cost feedback. This method optimizes long-term system costs through policy gradient updates, enabling intelligent learning of task offloading decisions.
- *TD3-based Computation Offloading Algorithm (TD3CO)*: TD3CO learns continuous offloading decisions using the TD3 method. By introducing double critic networks and delayed update mechanisms, it effectively alleviates the overestimation problem of the value function. This method optimizes offloading decisions in continuous action spaces, aiming to balance latency and energy consumption among multiple agents.

C. Simulation Results



(a) Energy cons. with USVs (b) Energy cons. with tasks



(c) KACO path planning (d) Energy cons. distribution

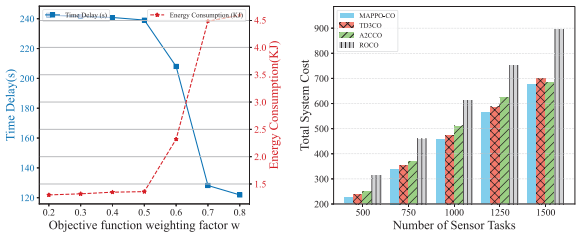
Fig. 3: Performance investigation with involved parameters.

Fig. 3(a) compares the total energy consumption of different path planning algorithms under varying numbers of USVs. The KACO algorithm consistently achieves the lowest energy consumption across all USV configurations (approximately 70–100 KJ), significantly outperforming the Munkres, ACO, GA, and Greedy algorithms. This improvement is mainly attributed to the introduction of K-means clustering, which partitions monitoring points into compact spatial clusters and enables each USV to operate within a localized region, thereby reducing redundant travel and long-distance movements.

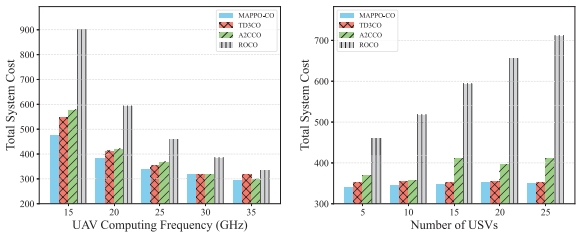
Fig. 3(b) compares the total energy consumption of different path planning algorithms under varying numbers of cluster points. As the number of cluster points increases from 500 to 1500, the total energy consumption of all algorithms shows an overall increasing trend. The Greedy algorithm consistently exhibits the highest energy consumption with the most significant growth, indicating its lack of global optimization capability in large-scale tasks. In contrast, the Munkres, ACO, and GA algorithms achieve relatively lower energy consumption, but their overall performance remains inferior to that of the KACO algorithm.

Fig. 3(c) illustrates the multi-USV path planning results achieved using the KACO algorithm. Within a 400 × 400 m^2 coordinate grid, the five USVs exhibit distinct trajectory plans. Notably, the path allocation of each USV is closely related to its energy consumption coefficient: USV 5, with a lower coefficient ($\eta = 9.81 J/m$), is assigned longer paths and more waypoints, whereas USV 2, with a higher coefficient ($\eta = 48.06 J/m$), follows shorter and more direct routes. This energy-driven path allocation mechanism effectively balances the overall system energy consumption, demonstrating the advantage of the KACO algorithm in optimizing energy efficiency.

Fig. 3(d) shows the energy consumption distribution of the five USVs under the KACO planning. The results indicate that the energy consumption among the USVs is relatively balanced (USV 1: 20.5%, USV 2: 22.4%, USV 3: 24.1%, USV 4: 12.4%, USV 5: 20.6%), despite the significant differences in their energy consumption coefficients. This balanced distribution confirms the effectiveness of the KACO algorithm in achieving energy equilibrium, ensuring that no single USV bears an excessive energy burden and that the overall system operates efficiently.



(a) Impact of weight factor (b) System cost with task loads



(c) UAV CPU frequency on system cost (d) USV quantity on system cost

Fig. 4: Performance comparison with different approaches.

Fig. 4(a) 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 UAVs 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. Hence, the analysis mainly focuses on the region where the weight factor remains moderate, ensuring a more balanced and stable trade-off.

Fig. 4(b) illustrates the trend of total system cost as the number of sensor tasks increases under different optimization methods. As the number of tasks rises from 500 to 1500, the total system cost calculated by each algorithm shows an upward trend, which aligns with expected results. The figure shows that compared to baseline algorithms, the proposed MAPPO-CO algorithm demonstrates better cost control with the increasing task loads. The total system cost under MAPPO-CO grows more steadily with the number of tasks, indicating that its optimization strategy exhibits strong adaptability across

Fig. 4(c) compares system cost under different UAV computing frequencies. As frequency increases, all algorithms show reduced cost, consistent with the expectation that better

hardware lowers computation overhead. The results show that MAPPO-CO consistently achieves the lowest cost, demonstrating superior performance and adaptability across both low and high-frequency settings.

Fig. 4(d) shows the total system cost with varying numbers of USVs. While the costs of ROCO and A2CCO rise significantly as the USV count increases, MAPPO-CO and TD3CO maintain relatively stable performance. In particular, MAPPO-CO consistently achieves the lowest cost across all scenarios, with only slight fluctuations as the number of USVs grows. This demonstrates the robustness and adaptability of MAPPO-CO in multi-USV environments.

VII. CONCLUSION

To enable efficient monitoring in urban water environments, we propose an intelligent system in which multiple USVs and UAVs collaboratively perform data collection and task offloading. The system combines energy-aware multi-USV path planning with a UAV-assisted edge computing framework using deep learning–based task offloading, jointly minimizing delay and energy consumption. Simulation results demonstrate that the proposed KACO and MAPPO-CO methods consistently outperform baselines in adaptability, efficiency, and stability. Future work will extend the system to real-world aquatic environments, enabling diverse tasks, large-scale multi-agent collaboration, and robust control under dynamic conditions.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (No. 62476276), 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). Shuo Xiao is the corresponding author.

REFERENCES

[1] T. Cattai, S. Colonnese, D. Garlisi, A. Pagano, and F. Cuomo, "Graphsmart: A method for green and accurate iot water monitoring," *ACM Trans. Sens. Networks*, vol. 20, no. 6, pp. 130:1–130:32, 2024.

[2] M. Jahanbakht, W. Xiang, N. J. Waltham, and M. R. Azghadi, "Distributed deep learning and energy-efficient real-time image processing at the edge for fish segmentation in underwater videos," *IEEE Access*, vol. 10, pp. 117 796–117 807, 2022.

[3] B. Wang, K. Ben, H. Lin, M. Zuo, and F. Zhang, "Ep-adta: edge prediction-based adaptive data transfer algorithm for underwater wireless sensor networks (uwsns)," *Sensors*, vol. 22, no. 15, p. 5490, 2022.

[4] Z. Wang, J. Du, C. Jiang, Y. Ren, and X.-P. Zhang, "Uav-assisted target tracking and computation offloading in usv-based mec networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 389–11 405, 2024.

[5] Z. Lv, X. Wang, G. Wang, X. Xing, C. Lv, and F. Yu, "Unmanned surface vessels in marine surveillance and management: Advances in communication, navigation, control, and data-driven research." *Journal of Marine Science & Engineering*, vol. 13, no. 5, 2025.

[6] C. Tang, T. Yao, S. Xiao, H. Wu, and R. Li, "Uav-usv collaborative task offloading for edge computing enabled smart lake monitoring," *J. Syst. Archit.*, vol. 176, p. 103830, 2026.

[7] M. Dai, N. Huang, Y. Wu, L. Qian, B. Lin, Z. Su, and R. Lu, "Latency minimization oriented hybrid offshore and aerial-based multi-access computation offloading for marine communication networks," *IEEE Transactions on Communications*, vol. 71, no. 11, pp. 6482–6498, 2023.

[8] X. Liu, L. Ho, S. Bruneel, and P. Goethals, "Applications of unmanned vehicle systems for multi-spatial scale monitoring and management of aquatic ecosystems: A review," *Ecological Informatics*, vol. 85, p. 102926, 2025.

[9] F. Sotelo-Torres, L. V. Alvarez, and R. C. Roberts, "An unmanned surface vehicle (usv): Development of an autonomous boat with a sensor integration system for bathymetric surveys," *Sensors*, vol. 23, no. 9, p. 4420, 2023.

[10] J. McMahon and E. Plaku, "Autonomous data collection with dynamic goals and communication constraints for marine vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 3, pp. 1607–1620, 2022.

[11] M. Cheng, Q. Guan, F. Ji, Y. Huang, and T. Q. Quek, "Mobile relaying between usv and auv under fer constraints for underwater data transmission," *IEEE Communications Letters*, vol. 27, no. 12, pp. 3429–3433, 2023.

[12] L. Zhao and Y. Bai, "Data harvesting in uncharted waters: Interactive learning empowered path planning for usv-assisted maritime data collection under fully unknown environments," *Ocean Engineering*, vol. 287, p. 115781, 2023.

[13] J. Liu, S. Anavatti, M. Garratt, and H. A. Abbass, "Modified continuous ant colony optimisation for multiple unmanned ground vehicle path planning," *Expert Systems with Applications*, vol. 196, p. 116605, 2022.

[14] J. Zhuang, L. Long, L. Zhang, Y. Zhang, and X. Li, "Research on task allocation for multi-type task of unmanned surface vehicles," *Ocean Engineering*, vol. 308, p. 118321, 2024.

[15] B. Liu, S. Jin, Y. Li, Z. Wang, D. Zhao, and W. Ge, "An asynchronous genetic algorithm for multi-agent path planning inspired by biomimicry," *Journal of Bionic Engineering*, vol. 22, no. 2, pp. 851–865, 2025.

[16] M. H. J. Bahabadi, A. Mahdavi, and S. Khankalantary, "Heterogeneous coverage path planning for multi-agent systems with aco and ga," in *2024 32nd International Conference on Electrical Engineering (ICEE)*. IEEE, 2024, pp. 1–6.

[17] C. Yin, K. Shi, and H. Wang, "The equal-time waypoint method: A multi-auv path planning approach that is based on velocity variation," *Drones*, vol. 9, no. 5, p. 336, 2025.

[18] C. Li, K. Jiang, Z. Zhang, C. Xiong, and S. Wan, "Joint service caching and computation offloading scheme with 3d uav deployment for icvs in uav-assisted vec," *IEEE Transactions on Communications*, pp. 1–1, 2025.

[19] J. Tang, J. Nie, Y. Zhang, Z. Xiong, W. Jiang, and M. Guizani, "Multi-uav-assisted federated learning for energy-aware distributed edge training," *IEEE Transactions on Network and Service Management*, vol. 21, no. 1, pp. 280–294, 2023.

[20] M. Falcão, C. B. Souza, A. Balieiro, and K. Dias, "Resource allocation for uav-enabled multi-access edge computing," *The Journal of Supercomputing*, vol. 80, no. 15, pp. 22 770–22 802, 2024.

[21] Y. Liao, X. Chen, J. Liu, Y. Han, N. Xu, and Z. Yuan, "Cooperative uav-usv mec platform for wireless inland waterway communications," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 3064–3076, 2023.

[22] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483–502, 2002.

[23] H. Niu, Z. Ji, A. Savvaris, and A. Tsourdos, "Energy efficient path planning for unmanned surface vehicle in spatially-temporally variant environment," *Ocean Engineering*, vol. 196, p. 106766, 2020.

[24] T. Shen, J. Guo, H. Liang, Y. Li, K. Li, Y. Dai, and Y. Ai, "Research on a blue–green led communication system based on an underwater mobile robot," in *Photonics*, vol. 10, no. 11. MDPI, 2023, p. 1238.

[25] M. Stojanovic and J. Preisig, "Underwater acoustic communication channels: Propagation models and statistical characterization," *IEEE communications magazine*, vol. 47, no. 1, pp. 84–89, 2009.

[26] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Communications magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[27] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.